The Influence of Leadership on the Adoption of Agile Practices and Principles in Software Development Teams

by:

Alicia Maria Parkes

Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of the degree of

Doctor of Philosophy of Loughborough University

January 2020

Supervisors: Dr. Crispin Coombs

Prof. Guy Fitzgerald

© by Alicia Parkes, 2020

Acknowledgements

To the family, friends and colleagues who supported and encouraged me, I am eternally grateful. There were times when I wondered if there would ever be an end to this PhD, and your words of encouragement kept me going. I am delighted to be able to answer the question “is it done yet”, finally, with – Yes!

Special thanks go to my parents, who took me back in ‘for a few months’ so I could focus on writing up my thesis and didn’t complain when I was still there almost a year later. I could not have done this without your help and support. Thank you.

I must also thank each and every individual who participated so willingly in my research. Thank you not only for your time but also your openness in sharing your experiences and thoughts with me. I hope that I have done them justice.

Particular thanks and gratitude go to my supervisors, Dr. Crispin Coombs and Prof. Guy Fitzgerald. This research has taken 6 long years to complete, and I am sure, like me, you wondered if it would ever actually end! I can honestly say that without the vast amounts of support, advice and feedback you offered me, this thesis would not have been possible. You have been incredibly generous with your time and I could not have asked for better supervisors.

Additional thanks go to Prof. Fitzgerald, who despite ‘technically’ retiring in the middle of my PhD, still diligently provided support and advice. Thank you for sticking it out with me, sorry it took me so long!

Abstract

With the growth of globalisation and significant developments in science and technology, software development today requires the capability to deal with fast-paced changes, increased competition and higher customer expectations. In light of this, many organisations have moved away from traditional approaches to software development, preferring instead to adopt lightweight, iterative approaches commonly termed ‘agile methodologies’.

Despite the popularity of agile practices and principles, adoption rates and success in using them vary. Few academic studies exist examining how agile practices and principles are adopted in practice, or why there is such variation in their use. This study addresses this gap by drawing on transformational-transactional leadership theory to understand how agile practices and principles are adopted, and what factors influence this adoption. It also draws on existing research highlighting a distinction between cognitive and behavioural change and applies this to agile adoption.

A qualitative approach with an interpretive perspective was taken for this research, adopting a case study research method. A single case, SoftwareCorps, was chosen with data collected over three and a half years, predominantly through participant observation and two phases of semi-structured interviews. Overall, 29 individuals participated in 47 interviews.

This study was undertaken in two phases. In the first phase, the journey of agile adoption at SoftwareCorps was explored, and factors that influenced agile adoption attempts considered. In the second phase, the focus was narrowed to explore how leadership influenced the adoption of agile practices and principles. A model of leadership for agile adoption was subsequently developed, combining leadership theory and insights from the empirical data in this study.

Findings from phase one demonstrated that effective and open communication, coaching, understanding the existing system of working, removal of barriers, sponsorship and buy-in, piloting with volunteers, selection of Scrum, and leadership were all influential factors in the adoption of agile at SoftwareCorps. However, leadership was found to be the most dominant theme. Additionally, data analysis in phase one revealed a distinction between behavioural and cognitive changes required for the adoption of agile practices.

Findings from phase two revealed that a leader’s interpretation of agile and style of leadership shaped the type of change (i.e. behavioural or cognitive change) driven in software development teams at SoftwareCorps. A transformational leadership style, combined with an interpretation of agile as an approach to problem solving and a drive to create cognitive change, was found to be conducive to successful agile adoption. A transactional leadership style with an interpretation of agile as a process and a focus exclusively on behavioural change was inhibiting to agile adoption.

The findings of this study suggest that successful agile adoption requires software development teams to be supported by individuals whose leadership style compliments agile values and principles (such as empowering, communicating and collaborating). Additionally, successful agile adoption is facilitated by leaders whose interpretation of agile is as an approach to problem-solving, rather than merely a set of procedures for developing software.

This study contributes towards a greater understanding of factors influential to the adoption of agile practices and principles and provides a model to explain how leadership may influence agile adoption. In so doing, this study helps to explain why variation in agile adoption levels may occur.

Table of Contents

1 Introduction 1

1.1 Research Background 1

1.1.1 How does agile address the problem? 2

1.2 Research Aim 4

1.3 Research Context 6

2 Literature Review – Agile Software Development 9

2.1 Software Development Methodology 9

2.1.1 Use of Systems Development Methods in Practice 10

2.1.2 Evolution of Software Development Methodologies 13

2.2 Agile Software Development 16

2.3 Defining Agile Software Development 19

2.3.1 Agile Methods 21

Extreme Programming (XP) 21

Feature Driven Development (FDD) 22

Dynamic System Development Method (DSDM) 22

Adaptive Software Development (ASD) 22

Crystal 23

Lean Development 23

Kanban 23

Rational Unified Process (RUP) 24

Scrum 24

Summary 27

2.3.2 Agile principles and practices 27

Categorisation 41

Interpretation 42

2.4 Perspectives on Agile Software Development 43

2.4.1 Themes in the Agile Literature 49

IT-enabled agility 49

Project Performance 50

Agile Usage 51

2.4.2 Adoption of Agile Working Practices 53

Factors influencing adoption of working practices 57

Behavioural vs. cognitive adoption 62

2.4.3 What theory has been used to study agile 65

2.4.4 What research methods are used most? 67

2.5 Summary 68

3 Literature Review - Leadership 71

3.1 Defining Leadership 71

3.2 Perspectives on Leadership 74

3.3 Leadership in the IS Domain 75

3.4 Leadership Theory 81

3.4.1 Transformational and Transactional Leadership Theory 83

Transactional Leadership Characteristics 87

Transformational Leadership Characteristics 89

Individual Consideration 90

Intellectual Stimulation 91

Inspirational Motivation 92

Idealised Influence (Charisma) 93

Agile and Transformational Leadership 96

3.5 Summary 97

4 Research Methodology 101

4.1 Research Aim 101

4.2 Research Philosophy 102

4.2.1 Interpretivist Perspective 103

4.3 Research Approach 103

4.3.1 Quantitative 104

4.3.2 Qualitative 104

4.3.3 Mixed-Methods 105

4.3.4 Choice of Research Approach 105

4.4 Research Strategy 105

4.4.1 Case Study 106

4.4.2 Action Research 106

4.4.3 Choice of Research Method 107

4.5 Research Design 107

Case study questions 108

Case study proposition 108

Unit of analysis 108

Linking data to proposition 109

Criteria for interpretation of findings 109

4.5.1 Credibility 109

Validity 110

Reliability 112

4.5.2 Ethics 112

4.5.3 Identify a Research Problem 113

4.5.4 Selection of the Case 114

Selection of the Participants 114

4.5.5 Data Collection Methods 116

Interviews 117

Participant Observation 117

Documents 118

4.5.6 Data Collection Process 118

Pilot Interviews 118

Semi-Structured Interviews 119

Designing the interview instruments 119

Participant Observation 121

Data Collection Log 121

4.5.7 Validity and Triangulation 122

4.6 Data Analysis 123

4.6.1 Transcription 123

4.6.2 Coding 124

Coding Strategy 125

Codes-to-Theory 126

4.6.3 Data Display 127

Emergence of the Leadership Theme 129

4.6.4 Write-up 132

4.7 Summary 132

5 A History of SoftwareCorps and their Agile Journey 135

5.1 Summary of the Participants and Key Roles 135

5.2 Agile Journey at SoftwareCorps 137

5.2.1 Pre-agile (1993-2008) 139

5.2.2 Agile-but (2008-2010) 141

5.2.3 Adaptation of Agile (2010-2013) 145

5.2.4 Introducing Agile by the book (2013-2014) 149

5.2.5 Embedding Agile by the book (2014-2016) 152

5.2.6 Return to Command and Control (2016-2017) 156

6 Phase One Findings – Agile Principles 161

6.1 Introduction 161

6.2 Self-Organising Teams 165

6.2.1 Practices associated with Self-Organising Teams 165

6.2.2 Adoption of Self-Organising Teams 167

6.3 Deliver Working Software Frequently 169

6.3.1 Practices associated with Delivering Working Software Frequently 169

6.3.2 Adoption of Delivering Working Software Frequently 170

6.4 Reflect and Improve 172

6.4.1 Practices associated with Reflect and Improve 173

6.4.2 Adoption of Reflect and Improve 173

6.5 Cross-Functional Teams 175

6.5.1 Practices associated with Cross-Functional Teams 175

6.5.2 Adoption of Cross-Functional Teams 177

6.6 Satisfy the Customer 180

6.6.1 Practices associated with Satisfy the Customer 180

6.6.2 Adoption of Satisfy the Customer 181

6.7 Welcome Changing Requirements 183

6.7.1 Practices associated with Welcome Changing Requirements 183

6.7.2 Adoption of Welcome Changing Requirements 184

6.8 Face to Face Communication 185

6.8.1 Practices associated with Face to Face Communication 185

6.8.2 Adoption of Face to Face Communication 186

6.9 Promote Sustainable Development 187

6.9.1 Practices associated with Promote Sustainable Development 187

6.9.2 Adoption of Promote Sustainable Development 187

6.10 Motivated Individuals 189

6.10.1 Practices associated with Motivated Individuals 189

6.10.2 Adoption of Motivated Individuals 189

6.11 Technical Excellence and Good Design 190

6.11.1 Practices associated with Technical Excellence and Good Design 190

6.11.2 Adoption of Technical Excellence and Good Design 192

6.12 Simplicity is Essential 193

6.12.1 Practices associated with Simplicity is Essential 194

6.12.2 Adoption of Simplicity is Essential 194

6.13 Working Software is Primary Measure of Progress 195

6.13.1 Practices associated with Working Software is Primary Measure of Progress 195

6.13.2 Adoption of Practices associated with Working Software is Primary Measure of Progress 196

6.14 Context 197

6.14.1 Agile outside of Software Development 197

6.14.2 Motivation for introduction of agile 199

6.15 Factors influencing the adoption of working practices associated with Agile Principles 201

6.15.1 Effective and Open Communication 202

Leaders Influence 203

6.15.2 Coaching 203

Leaders Influence 204

6.15.3 Understanding Existing System 205

Leaders Influence 205

6.15.4 Removal of Barriers 206

Leaders Influence 207

6.15.5 Sponsorship & Buy-in 207

Leaders Influence 208

6.15.6 Piloting with Volunteers 208

Leaders Influence 209

6.15.7 Selecting Scrum 209

Leaders Influence 210

6.16 Leadership and Agile Adoption 211

6.17 Summary of Phase One Findings 211

7 Phase Two Findings - Leadership 215

7.1 Leadership 216

7.1.1 Nelson’s Leadership Style 217

7.1.2 Nelson’s interpretation of agile and resulting impact on working practices 221

7.1.3 Julius’ Leadership Style 227

7.1.4 Julius’ interpretation of agile and resulting impact on working practices 231

7.1.5 Summary of Leadership Theme 236

7.2 Relationships between Leadership and Agile Working Practices 240

7.2.1 Relationship between Leadership and Self-Organising Teams 240

Participants’ Understanding of Self-organisation 241

Individual Freedom 242

Individual Freedom Under Nelson 243

Individual Freedom Under Julius 244

Summary 245

Team Freedom 245

Team Freedom Under Nelson 245

Team Freedom Under Julius 246

Summary 247

Organisational Freedom 248

Organisational Freedom Under Nelson 248

Boundaries to Organisational Freedom Under Nelson 250

Organisational Freedom Under Julius 251

Summary 252

Summary of Relationship Between Leadership and Self-organising Teams 253

7.2.2 Relationship between Leadership and Reflect and Improve 256

Participants’ Understanding of Reflect and improve in the context of agile 256

Reflect and Improve under Nelson’s leadership 257

Boundaries to Reflect and Improve under Nelson 259

Reflect and improve under Julius’ Leadership 260

Summary of the relationship between Leadership and Reflect and Improve 261

7.3 Summary of Findings 264

8 Discussion 271

8.1 Introduction 271

8.2 RQ1: How do the practices of agile software development enable the manifestation of agile principles in software development teams? 272

8.2.1 Behavioural and Cognitive Adoption 274

8.2.2 What factors influence the adoption of agile working practices in software development teams? 278

8.3 RQ2: How does leadership influence the adoption of agile working practices in software development teams 287

Leader’s Interpretation of Agile 288

Leadership Style 289

8.4 A conceptual model of leadership for agile adoption 293

Leadership for Agile Adoption 293

Leader’s interpretation of Agile 293

Leadership Style 296

Change Type 297

Team Behaviours 298

Agile adoption outcome 299

8.4.1 Paths through the Conceptual Model of Leadership for Agile Adoption 303

Nelson’s Journey 303

Julius’ Journey 305

8.4.2 Summary 306

9 Conclusion 307

9.1 Contributions to Theory 307

9.2 Contributions to Practice 309

9.3 Limitations and Recommendations for Future Research 310

9.4 Concluding Remarks 313

10 References 315

11 Appendix 339

11.1 Email inviting participants to interview 340

11.2 Informed Consent and Demographics Form 341

11.3 Interview Protocol 342

11.4 Overview of Findings in Leadership Literature 348

11.5 Summary of definitions of the 4 I's of Transformational Leadership 350

11.6 Timeline of Software Development at SoftwareCorps 359

11.7 Network Model of Agile Evolution at SoftwareCorps 360

11.8 Matrix of the Participants’ Views of Agile Principles 361

Summary of Figures

Figure 1: Evolution of Agile Methods (Abrahamsson et al. 2003) 15

Figure 2: Ad-hoc Leadership Model by Dubinsky and Hazzan (2010) 80

Figure 3: A Priori Codes 125

Figure 4: Codes-to-Theory Model taken from Salanda 1987, p.12 127

Figure 5: Components of Data Analysis: Interactive Model (Miles et al. 2014 p.14) 128

Figure 6: Emergence of Leadership from Coding 130

Figure 7: Timeline of Agile journey at SoftwareCorps 138

Figure 8: Development Process v2 (DP2) 142

Figure 9: Software Development Teams 144

Figure 10: Project Framework 147

Figure 11: Organisational changes introduced by Julius 157

Figure 12: Conceptual Model of Leadership for Agile adoption 216

Figure 13: Roadmap Board 225

Figure 14: Problem Management Board 227

Figure 15: Differing levels of self-organisation 254

Figure 16: Leadership Style 255

Figure 17: Leader's Interpretation of Agile 255

Figure 18: Leadership and Reflect and Improve 262

Figure 19: Change Type 263

Figure 20: Team Behaviours 264

Figure 21: Leader's Interpretation of Agile 294

Figure 22: Leadership Style 296

Figure 23: Change Type 297

Figure 24: Team Behaviours 298

Figure 25: Conceptual Model of Leadership for Agile Adoption 300

Summary of Tables

Table 1: Drivers for Agile Working 3

Table 2: Definition of Agility in the IS Literature 20

Table 3: Top agile practices according to Version One 2019 29

Table 4: List of Agile Working Practices 31

Table 5: Agile working practices mapped to principles 39

Table 6: Summary of Agile Literature 45

Table 7: Adoption Factors and Approach 54

Table 8: Model of Learning and Change adapted from Fiol and Lyles (1985) 64

Table 9: Summary of Perspectives on IS Leadership 76

Table 10: Definitions of Transformational and Transactional Leadership from IS Literature 85

Table 11: Terms defined through review of leadership literature 95

Table 12: Case Study Tactics for Four Design Tests (adapted from Yin, 2014, p. 45) 110

Table 13: Cross Section of Participants Interviewed 116

Table 14: Summary of the Participants and their Roles 136

Table 15: Data-Driven Ranking of Agile Principles 163

Table 16: Factors influencing the adoption of agile working practices 202

Table 17: Comparison of CTO's Nelson and Julius 237

Table 18: Summary of Findings Key Points 265

Table 19: Adoption factors identified at SoftwareCorps 279

Table 20: Incidents and Key Evidence 301

# Introduction

This section will introduce the research problem and why it is of interest to the academic world before presenting the research aims and questions. It will also provide context for the research problem along with some background information which aims to justify the significance of the study. Once this is established, the research method and findings of this study will be summarised.

## Research Background

In a world where everything is available at the touch of a screen, organisations are under immense pressure to deliver quality software. Over the past 50+ years, markets have changed dramatically with the growth of globalisation and significant developments in science and technology (Kose & Ozturk, 2014). It is becoming necessary for organisations to learn, and learn quickly, to deal with change. Those organisations who embrace rapid change and make it part of their culture will emerge with an essential competitive advantage in an ever more competitive environment (Ewenstein et al., 2015). Furthermore, it is not enough for organisations to adapt to change; they must embrace change and influence change in order to achieve sustainable success in the long term (Conboy & Fitzgerald, 2004).

Organisations such as Blockbuster, Kodak, Blackberry and Nokia have all been affected by fast-changing markets and technologies (Binns et al., 2014). In some cases, an inability to respond to change quickly has meant bankruptcy for organisations that once held prominence in their industry. When the market is rapidly changing, organisations must follow suit.

For many years traditional software development followed classical software engineering methods, which put a heavy emphasis on predicting and planning entire requirement sets up-front and managing escalating costs by eliminating change early. This approach led to an inability to respond later in the product lifecycle (Highsmith & Cockburn, 2001). Consequently, these methods became synonymous with bureaucracy, heavy corporate structure and an inability to keep up with the rapid pace of technological change. These constraints presented a significant issue for organisations in environments where responsiveness was vital for survival. In uncertain environments and with rapid changes in business and technology, software development teams needed to be able to respond quickly (Lohan et al., 2010).

### How does agile address the problem?

Agile methods were developed in response to classical software development approaches being unable to deliver software effectively. Problems with these methods included an inability to respond to change, poor communication, long development lifecycles and low-quality software (Cho, 2008). With an increasing demand for innovation and a growing need to re-humanise an environment focused on detailed plan-driven development, the software development community called for revolution (Boehm, 2002). The rise of what we now call agile software development methodologies was in answer to a call from businesses and developers alike *“asking for lighter weight along with faster and nimbler software development processes”* (Abrahamsson et al., 2003, p. 244).

Agile methods claimed to address issues inherent in traditional, plan-driven approaches by accelerating time to market, increasing quality and productivity, and enhancing flexibility (Campanelli & Parreiras, 2015). This was achieved through following the values and principles laid out in the Manifesto for Agile Software Development (Agile Manifesto) (Beck et al., 2001b), including being responsive to change, collaborating closely with the customer and iteratively delivering working software.

As a philosophy, agile encourages *embracing* change and is designed to create an environment in which fast-paced change becomes an advantage rather than something to mitigate against (Avison & Fitzgerald, 2006a). Agile methods seek to take a more realistic perspective of what can and cannot be controlled, providing a mechanism to deal with the latter (Maskell, 2001)*.* As part of this approach, businesses are encouraged to be open to experimentation and learning as the new norm, not just as a means to an end (Maskell, 2001).

Motivations for seeking agility cited across the literature (see Table 1) include an increased pressure to deliver at speed, rapidly changing business and technology landscapes and increasingly unpredictable global markets (Chakravarty et al., 2013; Lee et al., 2015; Lee & Xia, 2010; Nazir & Pinsonneault, 2012; Vidgen & Wang, 2009).

Table 1: Drivers for Agile Working

|  |  |
| --- | --- |
| **Source** | **Driver’s for Agile Working** |
| Lee et al. (2015) | Today’s competitive business landscape and the need for competitive success. |
| Chakravarty et al. (2013) | Hypercompetition brought about by globalisation and rapid changes in consumer preferences and economic cycles. |
| Nazir and Pinsonneault (2012) | Evolving in increasingly turbulent environments to respond to market threats and opportunities with speed. |
| Lee and Xia (2010) | The unprecedented rate of change in business and technology. |
| Vidgen and Wang (2009) | Unpredictable markets, changing customer requirements, pressures of ever shorter time-to- deliver, and rapidly advancing information technologies. |

Organisations today continue to deal with uncertainty such as volatile prices, trade wars, global competition, fickle consumers and new regulation (Tallon et al., 2019). In response to this, the agile market is flooded with consultancies, certifications, frameworks and experts which propagate the idea of ‘agile’ as the solution to these challenges. Still, research focusing on how agile practices are actually adopted and embedded in practice is scarce (Dikert et al., 2016). A problem Chakravarty et al. (2013) highlighted some years ago;

“The recipe for organizational success in modern business environments increasingly calls for agility as an important ingredient. Yet, we lack a comprehensive understanding of how organizations build and leverage agility for superior performance.”

There is a general tendency towards research seeking to test the validity and efficacy of agile approaches (Erickson et al., 2005), leaving the question of how to go about adopting these methods unanswered. There is, therefore, a need for more research providing insights into *how* agile software development practices are adopted, factors influential to adoption, and how these manifest in day-to-day software development activities.

## Research Aim

The overarching aim of this study is to understand how agile practices and principles are adopted, a topic which has lacked attention in the literature to date. Agile methods have continued to grow in popularity since the publication of the Agile Manifesto in 2001, which in part sought to re-humanise a bureaucratic and controlling software development industry. Almost two decades have passed since its publication, and the language, ceremonies and ideals of agile are now widespread, with many organisations purporting to follow some sort of agile approach. Despite this, there remains a lack of academic research regarding agile practice usage (Dikert et al., 2016) and guidance on the adoption of agile software development methodologies (Cram & Newell, 2016; Sidky et al., 2007), as will be discussed further in Chapter 2.

There is no widely acknowledged approach or method to rolling out ‘agile’ despite popular references to this being a revolution in software development (such as Jeff Sutherland (2014) claiming Scrum can deliver twice as much in half the time). While proponents of agile maintain its efficacy and benefits, some question its universal applicability (Cram & Newell, 2016; Highsmith et al., 2002). There is evidence of variability within and across organisations regarding the 'type' of agile adopted (Cockburn, 2002), from agile in name only, to 'agile by the book' (Cram & Newell, 2016). This variation continues regarding the extent to which agile is adopted, from organisation-wide transformations to the adoption of an agile approach on specific projects or teams. The puzzle is, therefore, if agile is so good, why do successful adoption rates vary?

In order to address this research aim, the following research question (RQ) was defined;

**RQ1:** How do the practices of agile software development enable the manifestation of agile principles in software development teams?

This research question was investigated in a single case organisation; SoftwareCorps. The study was undertaken in two phases. In the first phase, RQ1 was investigated through exploring the journey of agile adoption at SoftwareCorps, the factors that influenced agile adoption attempts, and practices participants associated with agile working. Through early analysis of the data collected a number of factors appeared to influence the adoption of agile practices. Of these, however, leadership emerged as significant. In particular, changes in leadership and the style of leadership appeared to be important, with multiple participants attributing different phases of agile adoption to the individual leading the department at the time. This observation led to the addition of the following research question;

**RQ2:** How does leadership style affect the adoption of agile working practices in software development teams?

In the second phase this additional research question was investigated. The focus was narrowed to explore how leadership influenced the adoption of agile practices and principles. In answering these questions, this study informs how agile may be adopted in software development teams and, explains why there may be variability in how it is adopted. Factors influential to the adoption of agile practices and principles will be identified and explored. Based on this analysis, advice for both academics and practitioners regarding the adoption of agile principles and practices are provided.

## Research Context

The aim of this study, as discussed above, is to gain a better understanding of how agile software development practices are adopted to inform adoption variability. Its purpose is not only to understand ‘how’ agile practices are adopted but also to understand what factors influence adoption, a subject lacking attention in the literature. To address this deficiency, an in-depth study of a single case (SoftwareCorps) is carried out exploring the adoption of agile software development practices, factors which influenced adoption, and how agile principles were manifest in the organisation under study. An interpretive perspective was chosen, which enabled a focus on the experiences and narratives of each participant. This approached enabled an in-depth exploration of agile adoption within a real-life context.

SoftwareCorps undertook several attempts at transitioning to agile working (see Chapter 5 - A History of SoftwareCorps and their Agile Journey), and as such were a rich case from which to explore the research objectives. Additionally, the researcher was already embedded within the organisation, having worked at SoftwareCorps for several years before the research began. The position of the researcher and the opportunity to explore an ongoing agile transition influenced the selection of SoftwareCorps as a case.

A two-phased approach was taken for this study. Phase one focused on the adoption of agile practices in software development teams, and phase two narrowed the focus to explore the influence of leadership on the adoption of these practices. Analysis of findings from both phases led to the development of a model of leadership for agile adoption, building on transformational-transactional leadership theory.

In both phases, participant observation and semi-structured interviews were the primary forms of data collection. In total, 29 individuals participated in 47 interviews, with each interview lasting an hour on average.

# Literature Review – Agile Software Development

This chapter focuses on positioning this research within the broader body of knowledge in the IS domain. It is structured as follows; First terms relating to software development (SD) methodologies are introduced. Next, an overview of the evolution of SD methodologies is given. Focus is then narrowed to concentrate on agile software development, its definition, and how it has been studied in the literature to date.

## Software Development Methodology

It is first necessary to clarify some terms and concepts which appear throughout this thesis. The central focus of this study is on the adoption of an agile software development methodology. As such, the concept of a 'software development methodology,' also referred to as information systems development (ISD) in the literature, will be discussed. This subject has been covered in detail by several eminent scholars in the information systems field, including Avison and Fitzgerald (2006a), Iivari and Maansaari (1998) and Hirschheim et al. (1995). A brief overview will be provided here as a foundation for this study.

According to Hirschheim et al. (1995), a software development methodology is formed of concepts, methods, beliefs, values, and principles whose purpose is to aid in creating a culture of reflection in order to learn systematically. Similarly, Avison and Fitzgerald (2006a) posit that a development methodology is more than a series of techniques and tools, and must also include the underlying theories and assumptions that have shaped the development of the methodology. They suggest that in comparison to a method, a 'methodology' is a broader concept, including philosophies and beliefs in addition to tools and procedures. Avison and Fitzgerald (2006a) subsequently define a software development methodology as follows;

"A systems development methodology is a recommended means to achieve the development, or part of the development, of information systems based on a set of rationales and an underlying philosophy that supports, justifies and makes coherent such a recommendation for a particular context. The recommended means usually include the identification of phases, procedures, tasks, rules, techniques, guidelines, documentation, and tools. They might also include recommendations concerning the management and organization of the approach and the identification and training of the participants."

This study adopts Avison and Fitzgerald’s (2006a) definition of methodology, positioning the use of terms such as philosophies, principles, and practices of agile within the broader concept of agile as a software development methodology. This definition is fitting given the Agile Manifesto is made up of a set of principles and values, as will be discussed in Section 2.2.

### Use of Systems Development Methods in Practice

The concept of an ‘information system’ is the application of IT in a social and organisational context (Angell & Straub, 1993). Leading up to the 1990’s, much of the research in the information systems domain had focused on methodologies, driven by a desire to systemise and spread ‘good practice’ in developing information systems (Avgerou & Cornford, 1993). Debate ranged from what defined a ‘good’ methodology, the choice of methodology, and the nature of a methodology (prescriptive versus tool-box versus philosophy) (Avgerou & Cornford, 1993). Fitzgerald (1996) and Truex et al. (2000) argued that by focusing on the evaluation and comparison of formalised methodologies, less well-understood but equally important areas were neglected. For example, arguments against the use of a formalised methodology (Fitzgerald, 1996, 1998; Truex et al., 2000), and the reality that in many cases practitioners were not using (or at least adhering completely to) formalised development methods at all (Fitzgerald, 1996; Russo & Stolterman, 2000). Fitzgerald (1996) sought, therefore, to ascertain whether in some circumstances systems development was beyond such formalised methods.

In a later paper Fitzgerald (1998) found that 60 percent of those surveyed were not using methodologies, and of those 79 percent reported they did not intend to adopt one. Iivari and Maansaari (1998) argued that (up until this point in the 1990’s), research on the actual use of systems development methods was lacking. They agreed with Fitzgerald (1996, 1998) that in those studies which did exist, evidence suggested that methods were used ‘weakly’ or not followed explicitly. Subsequently, research appears to have sought to address this gap in the literature, with scholars such as Russo and Stolterman (2000) citing ‘numerous studies’ of methodology usage. These studies supported earlier indications that methodologies were not followed explicitly (Fitzgerald, 1996, 1998; Russo & Stolterman, 2000). Further to this, Russo and Stolterman (2000) challenged academics to consider the underlying purpose of research into information systems methodologies. They called for in-depth studies of practice in order to create rich descriptions, and for the research community to consciously consider if their purpose is to inform the improvement of systems design within organisations, design practice in general, to consider what is best for the user, or what is best for society (Russo & Stolterman, 2000).

Introna and Whitley (1997) questioned the value in developing ‘complete’ methodologies when the evidence suggested practitioners would pick and choose the parts that suited them. They argued that a tacit understanding of the world was vital to making sense of methodology usage, and that without this understanding methodology usage was destined to fail. Consequently, they recommended ‘institutionalised apprenticeships’ for learning methodologies, in which would-be systems developers spend time learning the use of methods and tools in the context of information systems development. This was in contrast to the tradition of short training courses in a particular methodology. They also argued that rather than being the visible focus of attention, methodologies should be the invisible tools used for completing the task at hand. As such, Introna and Whitley (1997) support a view of methodologies in which improvisation is designed into the method.

Iivari and Maansaari (1998) went on to suggest that systems development methodologies were undergoing considerable change, with the emergence and early adoption of Object Orientated (OO) methods. Nandhakumar and Avison (1999) also cited the influence of OO and Computer Aided Software Engineering (CASE) tools on a new wave of methodologies. Fitzgerald (1998) suggested that the incumbent methodologies, derived from old organisational environments, required an overhaul in order to meet the needs of rapid systems development emerging in the climate of the late 1990’s.

As a consequence of what Avgerou & Cornford (1993) call ‘the methodologies movement’, a multitude of methodologies emerged for the development and management of information systems (Avison & Fitzgerald, 2006b). Angell & Straub (1993) suggested that in many cases businesses were encouraged to invest in information systems on the basis that ‘everybody else is doing it’ and were consequently compelled to follow suit or risk being at a competitive disadvantage. They argued that this led to organisations being burdened with information systems which increased bureaucracy without the benefit of serving any ultimate purpose. Worse still, the increase in the application and use of such systems increased the perceived need to develop methods to better develop and manage them (Angell & Straub, 1993). There was an inherent belief that in defining ‘tidy methods’, control over the application of IT could be achieved and as a consequence, control over organisations could be obtained (Angell & Straub, 1993). On a similar theme, Wastell (1996) suggested that methodologies simply provided organisational rituals behind which anxiety could be hidden, and a feeling of security and efficiency could be achieved. However through adhering to a formalised and structured approach, creative thinking was inhibited and learning processes were jeopardised (Wastell, 1996). This view was shared by Nandhakumar and Avison (1999) who called this the ‘fiction’ of methodology, where traditional methods present an image of control but are too mechanistic to be valuable in daily systems development activities. Angell & Straub (1993) argued that, rather than attempting to increase control over the uncontrollable, information systems and methods developed to manage them must aid in increasing flexibility and agility, and to facilitate learning and continuous improvement.

### Evolution of Software Development Methodologies

Avison and Fitzgerald (2006b) define four eras of software development methodologies; pre-methodology, early methodology, methodology, and post-methodology. They define the pre-methodology era, as the label suggests, as the early period of software development. During this time, around the 1960s, no explicit methodology for software development existed. The focus was reportedly on programming, and the needs of the users were secondary to overcoming the limitations of the technologies of the time (Avison & Fitzgerald, 2006b). A lack of customer focus and methodological definition led to applications being delivered late, above budget, and requiring much rework. In an attempt to address these failings, it was deemed necessary to bring structure and standards to the field. As a result, early methodologies were developed in the form of the Software Development Life Cycle (SDLC), also known as the Waterfall Model (Avison & Fitzgerald, 2006b).

Waterfall, prevalent during the 1970s and 1980s, was characterised by sequential, staged phases, detailed documentation, and structured approval and change control processes. Principles underlying the Waterfall methodology included low customer involvement, robust documentation, and sequential structure. Practices associated with these principles included the up-front design of systems, creation of detailed requirement specification documents, and following discrete phases of sequential development (design> development> testing> maintenance). There were several criticisms of this approach, including inflexibility in responding to changing needs and user dissatisfaction. Projects were also still being delivered late and over budget. In response to this, ‘the methodology era' emerged (around the 1990s) along with a proliferation of development methodologies (Avison & Fitzgerald, 2006b).

Avison and Fitzgerald (2006b) categorise these methodologies into two movements – those seeking to improve on the existing Waterfall Model and those claiming to be new methodologies. As can be seen from Figure 1 there is a rich history of methods, principles and schools of thought that formed a grounding from which agility and the agile manifesto would be built.

Abbas et al. (2008) provide a detailed history of the evolution of agile methods and its roots, in which they track the progression of software development methods from Waterfall through the Spiral Model, Computer Aided Software Engineering (CASE), Rational Unified Process (RUP), Rapid Application Development (RAD) and others towards what is now considered agile software development. Methods such as V-Model, Spiral Model and Rational Unified Process (RUP) retained a heavyweight, document and plan driven approach (Abbas et al., 2008). Others, such as RAD and Rapid Iterative Production Prototyping (RIPP), emphasised getting working software (prototypes) to customers early and formed the basis from which agile software development methods such as Dynamic Systems Development Methodology (DSDM) evolved (Abbas et al., 2008). As incremental and evolutionary development methodologies became popular, agile software development was born.

Figure 1, taken from Abrahamsson et al. (2003), illustrates a number of software development methods which contributed to the publishing of the Agile Manifesto in 2001 (through dotted lines). It also includes some more philosophical schools of thought which Abrahamsson et al. (2003) argue have contributed to the evolution of the agile methods shown. It would be possible to take this model further and include other non-software schools of thought such as lean manufacturing and systems thinking, both of which provide colour to the emergence of agility, however this is outside of the scope of this literature review.

A close up of a map

Description automatically generatedFigure 1: Evolution of Agile Methods (Abrahamsson et al. 2003)

## Agile Software Development

Providing a brief history of agile software development, both Strode (2016) and Jiang and Eberlein (2009) drew on the extant literature to give an overview of the evolution of agile methods. Strode (2016) stated that:

"Agile methods are founded on management ideas from new product development, theories of complex adaptive systems, observations on effective teamwork and communication, and improving product quality in software projects. These methods came to prominence in the late 1990s and are now well-accepted world-wide with an estimated adoption level of 50%."

Jiang and Eberlein (2009) noted that the term 'agile' could be traced back to lean manufacturing, which focused on reducing waste, learning, fast delivery, and empowering teams. Most notable in this regard is perhaps the Toyota Motor Corporation (Hines et al., 2004), with lean thinking seen in the automotive industry a far back as 1915 (Conboy & Fitzgerald, 2004). The Iacocca Institute is recognised as being one of the first to introduce the term 'agile manufacturing' into mainstream business literature in 1991 (Conboy & Fitzgerald, 2004).

Around the same period that agile manufacturing became mainstream (in the 1990s), it became apparent in the software development industry that there was a proliferation of software development methodologies. This period is termed the ‘methodology’ era by Avison and Fitzgerald (2003), who suggest method proliferation was in response to limitations of early methodologies in the late 1970s and early 1980s.

The Agile Manifesto was born out of a shared need to look “*for an alternative to documentation driven heavyweight software development processes*” (Highsmith, 2001). It was officially created in 2001 by a group of leaders in the agile software engineering industry (including Extreme Programming, Scrum, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development and Pragmatic Programming) who later came together to form the Agile Alliance (Lang et al., 2013). It was the publication of this manifesto that established agility within the software development community (Conboy & Fitzgerald, 2004). They settled on the term ‘agile’ as a replacement for the commonly used expression ‘lightweight’ as they felt this rang truer with the core values, and addressed the concerns of some that ‘lightweight’ gave the wrong connotation (Highsmith, 2001). Traditionally in competition or debate, this group of 17 practitioners came together to create the Agile Manifesto. Calling for a shift in the underlying philosophy of traditional ISD approaches, the Agile Manifesto embodies twelve guiding principles and four values (Lang et al., 2013) which state;

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.” (Beck et al., 2001b)

The following 12 principles were created by the group to elaborate and fulfil their stated values;

“Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.” (Beck et al., 2001a)

It is this set of values and principles that define agile software development methodologies and are the focus of this research. In the sections that follow, the values and principles of the Agile Manifesto will be explored in more detail, and the definitions used in the literature to define agile software development will be discussed.

## Defining Agile Software Development

Although the Agile Manifesto is focused on guiding practitioners towards ‘better’ ways of developing software (Beck et al., 2001b), there is also acknowledgement that software development exists within a wider organisational environment. Accompanying the online publication of the Agile Manifesto is a history of its creation by Jim Highsmith (2001). In this he urges the reader to consider *“software development, methodologies, and organisations, in new - more agile – ways*”. As such, ‘agile’ does not refer only to the activities of software development. It refers to the values and culture within which software development activities take place (Highsmith, 2001). The creators of the Agile Manifesto sought to build better organisational communities within which they would want to work, driven by their negative experiences of working with irrational demands imposed through corporate power structures (Highsmith, 2001). Highsmith (2001) argues that this *“isn’t merely a software development problem, it runs throughout Dilbertesque*[[1]](#footnote-1) *organisations”.* The values and principles of the Agile Manifesto, therefore, are not limited to software development. An agile approach to software development, as promoted by Beck et al. (2001b) cannot be removed from its context within the wider organisation. Consequently, the concept of ‘an agile approach’ stretches beyond software development and reaches across the organisation as a whole. Therefore, throughout this thesis the term ‘agile’ is used to refer not only to software development activities, but also the wider organisational environment within which software development takes place. As such, the terms agile, agility and agile software development may be used interchangeably.

Although the Agile Manifesto provides a set of values and principles for agile software development, these are some-what interpretive, and there is no absolute definition (Conboy & Fitzgerald, 2004). Therefore, scholars are left with the task of defining agile and agile software development in the context of their research. Table 2 provides a summary of agile definitions from several papers selected for this review.

Table 2: Definition of Agility in the IS Literature

| References | Definition of Agility |
| --- | --- |
| Campanelli and Parreiras (2015) | Handling unstable requirements, delivering working software in short time frames, with high quality, and under budget are the main characteristics of agile methods compared to traditional ones. Being agile is to be able to rapidly adapt to change in a flexible way. The capability is reflected by the attributes of flexibility, velocity, leanness, learning, and response to change |
| Chakravarty et al. (2013) | The ability to sense opportunities for competitive action and marshal the necessary resources to seize those opportunities with speed and surprise, create new or disrupt competitive advantage and be resilient and adaptive to threats |
| Lu and Ramamurthy (2011) | The ability to cope with rapid, relentless, and uncertain changes and thrive in an environment of continually and unpredictably changing opportunities |
| Conboy (2009) | The continual readiness of an ISD method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment |
| Chow and Cao (2008) | The word ‘‘agile’’ by itself means that something is flexible and responsive, so agile methods implies its ‘‘[ability] to survive in an atmosphere of constant change and emerge with success’’ (Anderson, 2004, p. xxviii)” |
| Maskell (2001) | Agility is the ability to thrive and prosper in an environment of constant and unpredictable change. Agility is not only to accommodate change but to also relish the opportunities inherent within a turbulent environment |

Definitions of agile found in the literature share several commonalities, with most focusing on agile in general, although a few focused on the team specifically (Campanelli & Parreiras, 2015; Lee & Xia, 2010). Common themes were sensing, responding to, and creating change, detecting and seizing opportunities, and being resilient to threats in an unpredictable environment. There were also themes of responding with speed, surprise, ease, dexterity, and flexibility. Only one paper in this review (Campanelli & Parreiras, 2015) defined agile in comparison to traditional methods. Vidgen and Wang (2009) (citing Baskerville (2006)) however, suggested that in an attempt to define agile *“authorities…often cast about for its opposite”,* placing the two approaches at opposite ends of a spectrum. Vidgen & Wang (2009) assert that such arguments reflect a lack of understanding of agile concepts and may lead to agile methods becoming nothing more than a set of steps followed indiscriminately of context.

Based on the values and principles of the Agile Manifesto and the definitions of agile in the literature reviewed, the definition of agile used in this study is as follows;

“Agility is the ability to thrive and prosper in an environment of constant and unpredictable change. Agility is not only to accommodate change but to also relish the opportunities inherent within a turbulent environment” (Maskell, 2001)

This definition applies not only to agile software development, but also to agile organisational as a whole.

### Agile Methods

There are a number of software development methods that are considered 'agile' to a greater or lesser extent. These include Extreme Programming (XP), Feature-Driven Development (FDD), Dynamic System Development Method (DSDM), Adaptive Software Development (ASD), Crystal, Lean Development (LD), Kanban, Rational Unified Process (RUP), and Scrum (Abrahamsson et al., 2002; Campanelli & Parreiras, 2015; Chow & Cao, 2008; Highsmith et al., 2002). The recommended practices, roles, and artefacts of these methods have significant overlap. This is unsurprising given they share common values and principles. Campanelli and Parreiras (2015) and Abrahamsson et al. (2002) define a number of these approaches as summarised below. As Scrum is the focus of this research, more detail will be given on this method.

#### Extreme Programming (XP)

A collection of practices taken from existing methods,in which the word extreme refers totaking common-sense principles to the extreme. It consists of five phases: Exploration, Planning, Iterations to Release, Productionising, Maintenance, and Death (Abrahamsson et al., 2002). There are also a set of defined roles: Programmer, Customer, Tester, Tracker, Coach, Consultant, and Manager (Big Boss) (Abrahamsson et al., 2002). XP aims to enable successful software development even when requirements are uncertain or changeable. Collective code ownership and pair programming are examples of practices which characterise XP (Abrahamsson et al., 2002).

#### Feature Driven Development (FDD)

An adaptive method for designing and building software, as it is not concerned with all phases of the development lifecycle, it is designed to work alongside other processes (Abrahamsson et al., 2002). It is composed of five sequential processes: develop an overall model, build a features list, plan by feature, design by feature, and build by feature. It classifies roles into; key, supporting, and additional. Key roles include project manager, chief architect, development manager, chief programmer, class owner, and domain experts (Abrahamsson et al., 2002). Small, dynamically formed feature teams and regular builds are examples of practices which characterise FDD.

#### Dynamic System Development Method (DSDM)

A framework for rapid application development (RAD) which proposes adjusting the scope of functionality to fit fixed time and resource constraints, rather than the traditional approach of fixing the scope and flexing time and resources (Abrahamsson et al., 2002). DSDM consists of five phases: feasibility study, business study, functional model iteration, design and build iteration, and implementation (Abrahamsson et al., 2002). The first two phases are done once, with the subsequent three phases repeated iteratively and incrementally (Abrahamsson et al., 2002). Empowered teams and frequent delivery of products are examples of practices which characterise DSDM.

#### Adaptive Software Development (ASD)

Seeks to address problems developing large and complex systems through iterative and incremental delivery and constant prototyping (Abrahamsson et al., 2002). It focuses on "balancing on the edge of chaos" and as such, aims to provide guidance but not be so prescriptive as to inhibit creativity (Abrahamsson et al., 2002). Practices such as iterative development, feature-based planning, and customer reviews are examples of practices which characterise ASD. It is, however, one of the methods with the fewest prescribed practices.

#### Crystal

A set of methodologies from which organisations select the one most appropriate based on project size and criticality, applying method tailoring principles provided with the method (Abrahamsson et al., 2002). The tailoring method is based on contingency theory (Campanelli & Parreiras, 2015). Each method is denoted with a colour to indicate the ‘heaviness’ of the process with darker colours indicating more rigour. Crystal methodologies are time-boxed and focus on communication and collaboration (Campanelli & Parreiras, 2015). Practices involved in crystal vary based on which method is selected, although incremental delivery is central to them all (Abrahamsson et al., 2002).

#### Lean Development

Lean development is the application of lean principles, which originate from the manufacturing industry, to software development (Poppendieck & Poppendieck, 2003, p. xxi). Fundamentally lean development seeks to remove activities that do not add value for the customer. Lean Development augments agile software development by applying well-known and accepted lean principles to software development, providing "thinking tools" to help translate lean principles into agile practices (Poppendieck & Poppendieck, 2003, p. xxiv). Practices include eliminating waste, amplifying learning, decide as late as possible, delivery as fast as possible, empower the team, build in integrity and see the whole (Poppendieck & Poppendieck, 2003, p. xxv).

#### Kanban

Also originates from the lean manufacturing industry and was subsequently applied to software development (Campanelli & Parreiras, 2015). It focuses on visualising and managing flow, limiting work in progress, making policies explicit, and improving collaboratively (Wang et al., 2012). A central practice in Kanban is the use of a physical board to represent the system, and cards to represent flow through the system.

#### Rational Unified Process (RUP)

Despite not being considered as particularly agile, there have been suggestions that parts of the process might be applied to agile projects (Abrahamsson et al., 2002). It is a phased process for object-oriented system development, created to complement UML (Abrahamsson et al., 2002). RUP consists of four phases: inception, elaboration, construction, and transition (Abrahamsson et al., 2002). Visually modelling (UML), iterative development, and controlling software changes are examples of practices that characterise RUP.

#### Scrum

A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value (Schwaber & Sutherland, 2017). Based on empiricism, Scrum focuses on knowledge, experience, and decision making based on what is known (Schwaber & Sutherland, 2017). It is an iterative approach utilising timeboxed periods to set a cadence for incremental delivery of value and continuous improvement.

There are three roles within the scrum team, which is self-organising and cross-functional; the product owner, scrum master, and development team. The following definitions are based on those from the Scrum Guide (Schwaber & Sutherland, 2017).

The **product owner** is responsible for maximising the value of the product resulting from the work of the development team. The product owner is a single person, not a committee, and is responsible for prioritisation of the backlog.

The **development team** are the professionals who carry out the work to create a potentially releasable increment of the product each sprint. There are no titles or sub-teams within the development team. Although team members may have specialised skills, accountability belongs to the entire team.

The **scrum master** is responsible for supporting Scrum as defined in the Scrum Guide by helping everyone understand the theory, practices, rules, and values of Scrum. The scrum master is a servant leader and helps those outside of the team understand which of their interactions with the scrum team are helpful and which are not. They help change these interactions to maximise the value created by the team.

In addition to these roles, Scrum prescribes a set of events which punctuate its iterative cadence;

The **sprint** is a timebox of one month or less during which a "done", usable and potentially releasable product increment is created. Sprints are a fixed and consistent duration; the next sprint follows directly on from the last. Each of the scrum events described below is contained within the sprint.

**Sprint planning** is an event during which the scrum team collaboratively plan the work to be carried out during the sprint. An output of sprint planning is the agreement of a sprint goal that can be met through the implementation of the selected work items (product backlog). The scrum team decides what can be delivered and how it will be achieved by the end of the sprint to meet this goal.

The **daily scrum** is a 15-minute time-boxed event for the development team. It is held every day of the sprint at the same time and place to minimise complexity. The development team uses the daily scrum to assess progress and plan their work for the day in order to maximise their chances of achieving the sprint goal.

The **sprint review** is held at the end of the sprint to inspect the completed product increment and adapt the upcoming product backlog items if necessary. The sprint review, sometimes called a demo, is attended by all stakeholders and is intended to elicit feedback and foster collaboration.

The **Retrospective** is an opportunity for the scrum team to inspect itself, identify potential improvements, and plan to implement them. It is scheduled between the sprint review and sprint planning.

Finally, the following artefacts accompany Scrum;

The **product backlog** is a prioritised list of everything that is known to be needed in the product. It is a single source of truth for requirements and is continuously changing in response to customer needs. It is visible to everyone in the organisation.

The **sprint backlog** is a subset of the product backlog and contains only those items to be done during the current sprint. It is visible to everyone in the organisation.

The **definition of done** is defined collaboratively by the scrum team to ensure everyone understands what "done" means. It includes everything to be done for a product increment to be considered potentially releasable, and all team members must follow it as a minimum.

The events and artefacts which define the Scrum methodology are also the practices which characterise Scrum.

#### Summary

According to Version One (2018), an organisation who publish a “state of agile” report every year based on surveying practitioners, the majority of organisations adopt Scrum (56%), ScrumBan (a blend of Scrum and Kanban - 8%), or a Scrum/XP hybrid (6%). 14% of those surveyed reported using multiple methodologies in a hybrid approach. Indeed, in the case under study, Scrum was the predominant agile methodology being applied with elements of Kanban and XP also used. In addition to providing a basic overview of existing agile methods and their associated practices, this review also provided terms of reference for this thesis.

The Agile Manifesto is a set of values and principles which were designed to shape the type of environments in which its creators desired to work. However, it does not go as far as prescribing specific practices; this is left to the various methods described in this section. What follows is an overview of popular agile practices and how they relate to agile principles, as well as the day-to-day practices of software development.

### Agile principles and practices

Poppendieck and Poppendieck (2003, p. xxiv) define principles as guiding ideas and insights, and practices as the actions taken to enact those principles. They note that principles are universal but not always easy to apply, whereas practices give specific guidance on what to do. As Poppendieck and Poppendieck (2003, p. xxiv) argue, problems that develop from applying metaphors from other disciplines to software development may arise as a consequence of transferring their practices, rather than the guiding principles.

The practices that make up agile methods are based on the values and principles of the Agile Manifesto (Campanelli & Parreiras, 2015). In order for a process to be considered agile, it must reflect these agile principles and embody their essential characteristics (Sidky et al., 2007). Alaa and Fitzgerald (2013) argued, therefore, that agile practices should realise agile principles and Campanelli and Parreiras (2015) agree, stating that *"agile practices should help accomplish agile principles*." In the context of this study, centred on the adoption of agile software development, it is therefore essential to understand what is currently known regarding agile practice usage and how these practices contribute towards agile adoption.

Many scholars studying the adoption of agile working practices cite Version One's State of Agile Report (e.g. Alaa & Fitzgerald, 2013; Campanelli & Parreiras, 2015; Dikert et al., 2016; Tripp & Armstrong, 2018). This report is updated annually and includes an overview of the most commonly adopted agile practices globally. Dikert et al. (2016) were seen to refer to this report in order to gather data on large scale agile practice use, which they suggest is lacking in the academic literature. Given the use of this report in the academic literature, it was determined to be suitable as a source of agile practices for this study. In the most recent report (at the time of writing) the most commonly adopted agile practices were as follows;

Table 3: Top agile practices according to Version One 2019

|  |  |
| --- | --- |
| **Agile Practice** | **% Adoption\*** |
| Daily Stand-up | 86 |
| Sprint/Iteration Planning | 80 |
| Retrospectives | 80 |
| Sprint/Iteration review | 80 |
| Short iterations | 67 |
| Planning poker/team estimation | 61 |
| Kanban | 61 |
| Release planning | 57 |
| Dedicated customer/Product owner | 57 |
| Single team (integrated dev and test) | 54 |
| Frequent Releases | 50 |
| Common work area | 45 |
| Product road mapping | 45 |
| Story mapping | 38 |
| Agile portfolio planning | 33 |
| Agile/Lean UX | 28 |

\*Percentage adoption in Table 3 refers to the number of respondents to the VersionOne survey who indicated that they used a given practice.

This report from industry was supported by practices commonly referenced in the academic literature, for example Kurapati et al. (2012) also found stand-up/scrum meetings, sprint planning, sprint/iterations, short releases, retrospectives, single team, and sprint review meeting to be in the top agile practices reportedly used in agile organisations.

A core set of agile practices were curated for use in this study by combining practices most commonly referenced in the literature (Diebold & Dahlem, 2014; Fontana et al., 2014; McHugh et al., 2012; Sidky et al., 2007; Tripp & Armstrong, 2018; Williams, 2012) and those detailed in the Version One (2018) report. Table 4 provides a summary of agile working practices gathered from the academic literature. Practices that were similar but named differently were grouped (for example, 'sprint review' and 'demo', or 'common work area' and 'co-location') to provide a consolidated list of standard agile working practices from across the literature. The following sections discuss the different sources used to develop the curated list for this study.

Table 4: List of Agile Working Practices

| **Agile Working Practices** | **Sources** |
| --- | --- |
| Requirements as User Stories | Sidky et al. (2007), Williams (2012), Diebold & Dahlem (2014), Layman et al. (2004), Williams et al. (2010), Abbas et al.(2010), Buglione (2011), Soundararajan et al. (2012), VersionOne (2017) |
| Planning (games, meetings) | Sidky et al. (2007), Williams (2012), Diebold & Dahlem (2014), Layman et al. (2004), Williams et al. (2010), Abbas et al.(2010), Buglione (2011), Tripp & Armstrong (2018), VersionOne (2017) |
| Co-located in an agile physical environment | Sidky et al. (2007), Williams (2012), Layman et al. (2004), Abbas et al. (2010), Buglione (2011), Soundararajan et al. (2012) |
| Retrospectives | Sidky et al. (2007), Williams (2012), Diebold & Dahlem (2014), Williams et al. (2010), Soundararajan et al. (2012), Tripp & Armstrong (2018), VersionOne (2017) |
| Collaborating with customer, business and each other. Shared responsibility | Sidky et al. (2007), Williams (2012), Diebold & Dahlem (2014), Williams et al. (2010), Abbas et al. (2010), Buglione (2011) |
| Continuous integration | Sidky et al. (2007), Williams (2012), Diebold & Dahlem (2014), Williams et al. (2010), Tripp & Armstrong (2018), VersionOne (2017) |
| Frequent small releases | Sidky et al. (2007), Williams (2012), Diebold & Dahlem (2014), Layman et al. (2004), Soundararajan et al. (2012), VersionOne (2017) |
| Refactoring | Sidky et al. (2007), Williams (2012), Diebold & Dahlem (2014), Soundararajan et al. (2012), Tripp & Armstrong (2018), VersionOne (2017) |
| Agile estimation (planning poker, team estimates) | Sidky et al. (2007), Williams (2012), Williams et al. (2010), Soundararajan et al. (2012), VersionOne (2017) |
| Automated testing | Sidky et al. (2007), Williams (2012), Layman et al. (2004), Tripp & Armstrong (2018), VersionOne (2017) |
| Coding Standards | Sidky et al. (2007), Williams (2012), Abbas et al. (2010), Tripp & Armstrong (2018), VersionOne (2017) |
| Short iterations | Sidky et al. (2007), Williams (2012), Abbas et al. (2010), Soundararajan et al. (2012), VersionOne (2017) |
| TDD/BDD | Sidky et al. (2007), Williams (2012), Williams et al. (2010), Tripp & Armstrong (2018), VersionOne (2017) |
| Visible product backlog, roadmap and vision | Sidky et al. (2007), Williams (2012), Diebold & Dahlem (2014), Soundararajan et al. (2012), VersionOne (2017) |
| Daily stand-up | Williams (2012), Diebold & Dahlem (2014), Tripp & Armstrong (2018), VersionOne (2017) |
| Iterative testing and quality assurance | Williams (2012), Diebold & Dahlem (2014), Layman et al. (2004), Abbas et al. (2010) |
| Multidisciplinary team, dedicated PO, shared goal | Williams (2012), Diebold & Dahlem (2014), Williams et al. (2010), VersionOne (2017) |
| Pair Programming | Sidky et al. (2007), Williams (2012), Soundararajan et al. (2012), VersionOne (2017) |
| Progress monitoring | Sidky et al. (2007), Diebold & Dahlem (2014), Williams et al. (2010), Soundararajan et al. (2012) |
| Release planning | Williams (2012), Williams et al. (2010), Tripp & Armstrong (2018), VersionOne (2017) |
| Self-organising (task volunteering, autonomous, empowered and motivated teams) | Sidky et al. (2007), Diebold & Dahlem (2014), Williams et al. (2010), Soundararajan et al. (2012) |
| Technical Excellence (code reviews, short build time, manage source code, consider architecture) | Williams (2012), Williams et al. (2010), Abbas et al. (2010), Buglione (2011) |
| Agile Documentation (minimal) | Sidky et al. (2007), Williams (2012), Soundararajan et al. (2012) |
| Collective code ownership | Williams (2012), Williams et al. (2010), VersionOne (2017) |
| Configuration management | Sidky et al. (2007), Williams (2012), Buglione (2011) |
| Emergent, rather than up-front design | Sidky et al. (2007), Williams (2012), VersionOne (2017) |
| Knowledge sharing | Sidky et al. (2007), Diebold & Dahlem (2014), Soundararajan et al. (2012) |
| Maintain a sustainable pace, minimal overtime | Williams (2012), Layman et al. (2004), Soundararajan et al. (2012) |
| Sprint Demo | Williams (2012), Diebold & Dahlem (2014), VersionOne (2017) |
| Supportive People & Team characteristics - small teams, highly communicative, high competent individuals | Sidky et al. (2007), Williams (2012), Williams et al. (2010) |
| Timeboxing | Williams (2012), Diebold & Dahlem (2014), Williams et al. (2010) |
| Complete definition of done | Williams (2012), Buglione (2011) |
| Continuous Feedback (giving and receiving) | Sidky et al. (2007), Soundararajan et al. (2012) |
| Kanban | Williams (2012), VersionOne (2017) |
| Simple work and software design | Layman et al. (2004), Buglione (2011) |
| Team Burndown and Velocity | Williams (2012), Tripp & Armstrong (2018) |

McHugh et al. (2011; 2012) investigated how three popular agile practices (retrospectives, sprint planning, and the daily stand-up) contributed towards motivation and trust in agile teams. These practices were selected given their focus on people, communication, interaction, and teamwork. They noted that leadership in agile teams is shared, which requires a higher level of trust from managers whose roles are considerably changed in an agile environment. The need to build trust between the team and its leaders has significance for this study, which takes a leadership perspective on agile adoption.

Tripp and Armstrong (2018) focused on how an organisation’s motivations for agile impacted how they tailored and selected agile practices. They proposed that understanding an organisations motives for agile adoption and identifying which practices fit with these goals could increase adoption success. Identification of motives as a factor in the literature has significance for this study, because the organisation that was studied (Software Corps) had several attempts at adopting agile working practices, at different periods in time, with different motivations. Diebold and Dahlem (2014) similarly focused on the usage and tailoring of agile practices, seeking to understand how project type, domain, or process influenced practice adoption. They found that some 'universal' agile practices were widely adopted, such as customer involvement, daily stand-up, and timeboxing. However, in line with Tripp and Armstrong (2018), Diebold and Dahlem (2014) found that in most cases, agile methods were adopted only partially with practices applied based on the domain and process. These findings hold relevance for this study as they inform the discussion around what influences the adoption of agile practices.

Williams (2012) sought to determine how well the Agile Manifesto's 12 principles captured what was valued by practising software engineers. The study found that 'original' agile practices such as continuous integration and short iterations were often ranked highly by respondents, whereas emergent practices such as planning poker[[2]](#footnote-2) and Kanban were at the bottom of the list. Williams (2012) ordered the agile principles in terms of how supported they were in practice. The principles of S*atisfy the customer* and *deliver working software frequently* were found to be the most supported, whereas the principle of *self-organising teams* was the least supported. Despite this, Williams (2012) found that there was still overwhelming support for the original principles, including self-organisation, even ten years after they were published.

Fontana et al. (2014) took a slightly different perspective, focusing on how to measure the maturity of agile teams. They searched the literature to identify commonly used agile practices. Respondents were subsequently asked to rate each practice on a five-point scale from ‘no maturity’ to ‘very high maturity’. They found that practitioners perceived maturity as fostering subjective capabilities, such as; collaboration, communication, commitment, care, sharing, and self-organisation. There were similarities between studies by Fontana et al. (2014) and Sidky et al. (2007), who sought to define a 'disciplined' approach to the adoption of agile practices.

Sidky et al. (2007) sought to address what they described as a lack of scholarly guidance for organisations regarding a structured approach to agile adoption. Sidky et al. (2007) present the SAMI model, which encompasses five agile levels used to identify the potential of a project or organisation to be agile. Their four-stage adoption process sought to enable organisations to determine their readiness for agile adoption, and which agile practices should be introduced. As part of their study Sidky et al. (2007) carried out a mapping of agile practices to a consolidated list of agile principles and categorised them under five levels of maturity; Level 1: Collaborative (enhancing communication and collaboration), Level 2: Evolutionary (early and continuous delivery of value), Level 3: Effective (adopt engineering practices to increase quality), Level 4: Adaptive (respond to change in the process) Level 5: Encompassing (Establishing a vibrant environment to sustain agility).

Given agile practices are purported to be the manifestation of agile principles, a search of the literature was carried out to determine which agile practices mapped to the 12 agile principles. Although a number of scholars carried out mapping exercise of some sort, for example, Strode (2016) mapped agile practices to common project dependencies, Alaa and Fitzgerald (2013) mapped agile practices and principles to complex adaptive systems principles, and Fontana (2014) and Sidky et al. (2007) both mapped practices to levels of agile maturity, a comprehensive mapping of the most commonly adopted agile practices to the 12 agile principles could not be found in a single study. Consequently, this mapping exercise was carried out by the researcher for use in this study.

This activity was necessary as mappings found in the existing literature tended to focused on aspects of their use, such as project success and team performance, rather than a holistic view of commonly adopted agile practices and how these contributed to the manifestation of agile principles. In many cases, the 12 agile principles were also condensed or grouped into a shortened list of principles.

For example, Alaa and Fitzgerald (2013) reduced the agile principles in the agile manifesto to 10 agile development principles. They merged motivated individuals, sustainable development and self-organising teams into a single ‘People-Oriented/Empowered & Self-organizing Teams’ principle, and collaboration between stakeholders and face to face communication into a single ‘Communication, Interactions & Collaboration’ principle. Although they did not explain why they do this explicitly, it can be inferred from their discussion that these principles shared many common practices based on their mapping. For this study, the researcher chose to work with the agile principles as they are given in the agile manifesto, as this is the information available to practitioners. Additionally, combining or reducing the agile principles requires an interpretation of those principles by the researcher, which may have unduly influenced the participants’ interpretations.

Practices collated from the literature were mapped against the 12 agile principles, with many practices applying to more than one principle. This was done as follows; First, a broad search of the agile literature was carried out focusing on agile principles, agile practices and agile adoption. Papers which specifically listed agile practices were shortlisted to form a basis for the curated list of agile practices used for this mapping. Practices which were referred to using different terms in the literature were combined to ensure the working list of practices was manageable. For example, 'sprint review' and 'demo' were combined under a single practice: ‘demo’. From this, a ‘master list’ of agile practices based on the academic literature was finalised.

Second, this list was compared to the list of most commonly adopted agile practices from the VersionOne (2018) report, which is based on responses from practitioners, to ensure all practices listed were covered, which they were. This master list of practices was then mapped in a table against the 12 agile principles, as taken from the Agile Manifesto. Where the literature did not provide mappings, the researcher defined these based on her own experience.

The results are presented in Table 5, which also provides a ranking of practices based on the number of agile principles they support. It is perhaps no coincidence that the majority of practices which map to three or more principles were also found to be in Version One’s (2018) top practices, except for burndown & velocity, knowledge sharing, and automated testing.

A detailed account of how agile practices help to manifest agile principles day-to-day in the case under study will be presented in Chapter 6. However, in order to better explain Table 5, an overview how practices help to manifest the principle of Self-organising Teams will be given as part of this review.

As presented in Table 5, the principle of self-organising teams is supported by the following agile practices; co-located teams, burndown & velocity, Kanban, knowledge sharing, daily stand-up, collaborating and shared responsibility, supportive people and team characteristics, and collective code ownership. Misra et al. (2012) cite the flexibility in the roles and responsibilities of team members as central to self-organisation, suggesting this facilitates the innovation and the creativity of team members. As such they are empowered to address challenges as they occur in an agile and responsive way without the need to reach outside of the team. This relates to *collaborating and shared responsibility* and *supportive people and team characteristics* in Table 5. Misra et al. (2012) also note the importance of increasing the dynamics of interactions and communication between team members for self-organisation, relating to co-location in Table 5. Fontana et al. (2014) also highlighted collaboration with respect to self-organisation, and additionally called out collective code ownership as a facilitator to self-organisation.

In the following section, categorisation of common agile practices by scholars is reviewed. Variation in how practitioners interpret agile practices and principles, as discussed in the literature, is also presented. Both categorisation and interpretation of agile practices and principles were important considerations for this study, as will be discussed.

Table 5: Agile working practices mapped to principles

| **Agile Principles** | **Self-organisation** | **Deliver Working Software** | **Reflect & Improve** | **Cross-functional Teams** | **Satisfy the Customer** | **Welcome Changing Requirements** | **Face to Face Communication** | **Promote Sustainable Development** | **Motivated Individuals** | **Technical Excellence** | **Working Software** | **Simplicity is Essential** | **Count** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Agile Practices** |  |  |  |  |  |  |  |  |  |  |  |  |
| User stories |  | ✓ |  |  | ✓ | ✓ |  |  |  |  | ✓ | ✓ | **5** |
| Co-located | ✓ |  |  | ✓ |  |  | ✓ |  | ✓ |  |  |  | **4** |
| Burndown & velocity | ✓ |  | ✓ |  |  |  |  | ✓ |  |  | ✓ |  | **4** |
| Kanban | ✓ |  | ✓ |  |  | ✓ |  | ✓ |  |  |  |  | **4** |
| Demo |  |  | ✓ |  |  | ✓ | ✓ |  |  |  | ✓ |  | **4** |
| Frequent small releases |  | ✓ |  |  | ✓ |  |  |  |  |  | ✓ | ✓ | **4** |
| Knowledge sharing | ✓ |  |  | ✓ |  |  |  |  | ✓ |  |  |  | **3** |
| Automated testing |  | ✓ |  |  | ✓ |  |  |  |  | ✓ |  |  | **3** |
| Short iterations |  | ✓ |  |  | ✓ | ✓ |  |  |  |  |  |  | **3** |
| Multi-disciplinary team, dedicated PO, shared goal |  |  |  | ✓ | ✓ |  |  |  |  |  |  |  | **2** |
| Daily stand-up | ✓ |  |  |  |  |  | ✓ |  |  |  |  |  | **2** |
| Collaborating & shared responsibility | ✓ |  |  | ✓ |  |  |  |  |  |  |  |  | **2** |
| Supportive people & team characteristics | ✓ |  |  |  |  |  |  |  | ✓ |  |  |  | **2** |
| Collective code ownership | ✓ |  |  |  |  |  |  |  |  | ✓ |  |  | **2** |
| Timeboxing |  | ✓ |  |  |  |  |  | ✓ |  |  |  |  | **2** |
| Visible backlog & roadmap |  | ✓ |  |  | ✓ |  |  |  |  |  |  |  | **2** |
| Release planning |  | ✓ |  |  | ✓ |  |  |  |  |  |  |  | **2** |
| TDD/BDD\* |  |  |  |  | ✓ |  |  |  |  | ✓ |  |  | **2** |
| Continuous feedback |  |  |  |  | ✓ | ✓ |  |  |  |  |  |  | **2** |
| Pair programming |  |  |  |  |  |  | ✓ |  |  | ✓ |  |  | **2** |
| Self-organising | ✓ |  |  |  |  |  |  |  |  |  |  |  | **1** |
| Definition of done |  | ✓ |  |  |  |  |  |  |  |  |  |  | **1** |
| Retrospectives |  |  | ✓ |  |  |  |  |  |  |  |  |  | **1** |
| Emergent design |  |  |  |  |  | ✓ |  |  |  |  |  |  | **1** |
| Agile estimation |  |  |  |  |  |  |  | ✓ |  |  |  |  | **1** |
| Sustainable pace, minimal overtime |  |  |  |  |  |  |  | ✓ |  |  |  |  | **1** |
| Planning games |  |  |  |  |  |  |  | ✓ |  |  |  |  | **1** |
| Technical excellence |  |  |  |  |  |  |  |  |  | ✓ |  |  | **1** |
| Continuous integration |  |  |  |  |  |  |  |  |  | ✓ |  |  | **1** |
| Refactoring |  |  |  |  |  |  |  |  |  | ✓ |  |  | **1** |
| Iterative testing |  |  |  |  |  |  |  |  |  | ✓ |  |  | **1** |
| Coding standards |  |  |  |  |  |  |  |  |  | ✓ |  |  | **1** |
| Config. management |  |  |  |  |  |  |  |  |  | ✓ |  |  | **1** |
| Progress monitoring |  |  |  |  |  |  |  |  |  |  | ✓ |  | **1** |
| Agile documentation |  |  |  |  |  |  |  |  |  |  |  | ✓ | **1** |
| Simple work and design |  |  |  |  |  |  |  |  |  |  |  | ✓ | **1** |

#### Categorisation

Across the literature, several scholars have categorised agile practices in order to understand how they are used. Chow and Cao (2008), for example, used five categories of practices to help determine potential success factors: Organizational, People, Process, Technical, and Project. Campanelli and Parreiras (2015) used a simpler, although similar categorisation based on Lee and Yong (2013), opting for the categories of management (e.g. daily stand-up), software process (e.g. collective code ownership) and software development (e.g. pair programming).

In many cases, including those cited here, categorisation is done to identify commonalities across projects, teams or organisations, and to identify patterns in practice usage and success. In their search of the agile practice literature, Gupta et al. (2019) noted that there was a high degree of agreement and overlap between scholars regarding practice categorisation. Gupta et al. (2019) cited Hummel et al. (2015) to asserted the importance of social interactions, social behaviour, and communication in agile software development. Gupta et al. (2019) supported the theoretical construct of *'social' agile practices*, which foster interactions, collaboration, and direct communication, as well as the construct of *'technical' agile practices* for engineering orientated activities. They note that;

“it is important to view software development through the lens of technically and socially-oriented [agile software development] practices because focusing on either one will result in an incomplete evaluation of software development methodologies” (Gupta et al., 2019)

As discussed earlier in this section (see Table 5), a mapping exercise was carried out demonstrating how day-to-day agile practices mapped to the 12 agile principles. This mapping demonstrated how agile principles manifest in day-to-day software development activities.

Further categorisation of agile practices, based on categories discussed in this section, was also considered. However, given the focus of this research on understanding practices in the context of agile principles, it was determined by the researcher that additional categorisation added unnecessary complexity. Interest was taken in the distinction made in the literature between technically- and socially-orientated practices, which shared similarities with the cognitive and behavioural change literature. Cognitive and behavioural change will be discussed later in this chapter (Section 2.4.2).

#### Interpretation

As a set of 4 values upheld by 12 principles, translating the Agile Manifesto into practice is mostly down to interpretation, which inherently introduces variation in how it is manifest day-to-day. In their study of perceived barriers to effective knowledge sharing in agile software development teams, Ghobadi and Mathias (2016) acknowledged perceptual differences in how different roles interpreted agile principles and practices. For example;

“user representatives may expect developers to be constantly open to new system requirements (referring to the ‘welcome change’ agile principle), and thus they may interpret developers’ inflexibility in welcoming change as a barrier” (Ghobadi & Mathiassen, 2016)

They suggested this disparity was of particular prevalence in teams where individuals demonstrated extreme dedication in adhering to and interpreting agile principles, and that this could exacerbate communication difficulties between roles. As Dingsøyr et al. (2016) note concerning the principle of self-organisation, there is little guidance in the Agile Manifesto regarding how these principles translate into practice. Sidky et al. (2007) also note a lack of scholarly advice regarding a structured process guiding the adoption of agile practices. This lack of guidance leaves much down to the interpretation of those involved in the transition to agile working. This may explain why adoption levels are so variable between organisations and teams. Having identified interpretation as a theme in the literature, and as a consequence of observing variation in the interpretation of agile and agile working at SoftwareCorps, this topic was explored through participant interviews and became a central concern of the findings in this study.

In the following sections, perspectives and themes from the existing agile literature are discussed along with gaps in current knowledge regarding agile adoption. This provides the foundation from which the research methodology for this study will be discussed in Chapter 4.

## Perspectives on Agile Software Development

In framing this research, it is essential to understand what is known regarding the adoption of agile in the IS domain. This chapter draws on several literature reviews carried out by scholars on the subject of agile software development (Campanelli & Parreiras, 2015; Chuang et al., 2014; Dingsøyr et al., 2016; Dingsøyr et al., 2012; Dingsøyr & Lassenius, 2016; Tallon et al., 2019)

In their comprehensive review, Chuang et al. (2014) found that research on agile software development had continued to increase in popularity between 2001-2012 and that early research focused on the *adoption* of agile methods. However, particularly after the publication of the Agile Manifesto in 2008, they noted a shift towards post-adoption and usage of agile methods. Dingsøyr and Lassenius (2016) supported this, reporting a focus on practices such as test-driven development (TDD) and pair programming or whole methods such as XP and Scrum. In more recent years, Chuang et al. (2014) noted an increase in researchers studying distributed teams and seeking to understand how to scale agile to larger projects.

Dingsøyr and Lassenius (2016) reported that the quality of studies in agile software development had increased over time, which they suggested was a consequence of several special issues in journals and conferences focusing on agile development. Mishra et al. (2017) reviewed the topics covered over six years at an agile conference for practitioners. They compared topics covered to those in the academic literature and found that leadership, which is the focus of this study, and coaching/mentoring had received almost no attention in the academic domain despite being a consistent focus for practitioners.

Despite the increased popularity of agile software development as a research topic, Chuang et al. (2014) reported that it was under-researched in comparison to industrial studies. They also found that agile was not well represented in the AIS journal basket of eight. The most cited journals were IEEE Software, Computer, Information and Software Technology, and Journal of Systems and Software. Despite this, 18 papers from the AIS journal basket were found for this review, spanning from 2000-2019.

However, given the tendency for agile research to be found outside of these journals, a number of other journals were included in this literature review, where the topic was highly related to this research. Table 6 provides an overview of these papers, along with a summary of the journals, methods, theories, and perspectives they cover.

The rest of this section is structured as follows; First, high-level themes from within the literature are discussed before focus narrows to the adoption literature. Second, theories prevalent in the literature are presented, and finally, dominant methods used to study agile in the IS domain are discussed. To conclude this section, the gaps this study seeks to address are presented.

Table 6: Summary of Agile Literature

| **Author** | **Title** | **Journal** | **Method** | **Theory** | **Theme** |
| --- | --- | --- | --- | --- | --- |
| Tallon et al. (2019) | Information technology and the search for organizational agility: A systematic review with future research possibilities | JSIS | Literature Review | n/a | Lit Review |
| Cram and Newell (2016) | Mindful revolution or mindless trend? Examining agile development as a management fashion | EJIS | Qualitative | Management fashion, Mindful innovation | Adoption |
| Ghobadi and Mathiassen (2016) | Perceived barriers to effective knowledge sharing in agile software teams | ISJ | Qualitative | Organisational role theory | Knowledge Sharing |
| Lowry and Wilson (2016) | Creating agile organizations through IT: The influence of internal IT service perceptions on IT service quality and IT agility | JSIS | Quantitative | IT service climate construct, Contingency Theory | How IT enables agility |
| Tripp et al. (2016) | Job Satisfaction in Agile Development Teams: Agile Development as Work Redesign | JAIS | Quantitative | Contingency Theory | Job satisfaction |
| Lee et al. (2015) | How Does IT Ambidexterity Impact Organizational Agility? | ISR | Quantitative | Capability building, Ambidexterity | How IT enables agility |
| Chakravarty et al. (2013) | Information technology competencies, organizational agility, and firm performance: Enabling and facilitating roles | ISJ | Quantitative | Theory of multiple contingencies | How IT enables agility |
| Ramesh et al. (2012) | Ambidexterity in Agile Distributed Development: An Empirical Investigation | ISJ | Qualitative | Contextual ambidexterity | Distributed teams |
| Nazir and Pinsonneault (2012) | IT and Firm Agility: An Electronic Integration Perspective | JAIS | Theoretical development | Electronic integration perspective | How IT enables agility |
| Tallon and Pinsonneault (2011) | Competing Perspectives on the Link Between Strategic Information Technology Alignment and Organizational Agility: Insights from a Mediation Model | MISQ | Quantitative | Strategic IT alignment | How IT enables agility |
| Lu and Ramamurthy (2011) | Understanding the link between Information Technology Capability and Organizational Agility: An Empirical Examination | MISQ | Quantitative | Structuration theory, Complementarity perspective | How IT enables agility |
| Lee and Xia (2010) | Toward agile: An integrated analysis of quantitative and qualitative field data on software development agility | MISQ | Mixed-Method | Social identity theory, Self-categorization theory | Project performance |
| Vidgen and Wang (2009) | Coevolving Systems and the Organization of Agile Software Development | ISR | Qualitative | CAS theory | Adoption (enablers and inhibitors) |
| Hovorka and Larsen (2006) | Enabling agile adoption practices through network organizations | EJIS | Quantitative | Network organisation | Adoption (enablers) |
| Fitzgerald et al. (2006) | Customising agile methods to software practices at Intel Shannon | EJIS | Qualitative | Contingency theory | Adoption (tailoring) |
| Sambamurthy et al. (2003) | Shaping Agility through Digital Options: Reconceptualizing the Role of Information Technology in Contemporary Firms | MISQ | Theoretical development | Capability building perspective | How IT enables agility |
| Beynon-Davies et al. (2000) | A case study of a rapid application development project | ISJ | Qualitative | None | Usage |
| Cho (2008) | Issues and Challenges of Agile Software Development with Scrum | Issues in Information Systems | Qualitative | None | Usage |
| Dingsøyr and Lassenius (2016) | Emerging themes in agile software development: Introduction to the special section on continuous value delivery | Information and Software Technology | Review of three papers | None | Review |
| Dingsøyr et al. (2016) | Team Performance in Software Development: Research versus Current Advice | IEEE Software | Literature Review | None | Team Performance |
| Strode (2016) | A dependency taxonomy for agile software development projects | Information Systems Frontiers | Qualitative | Dependency coordination theory | Project performance, Adoption |
| Chow and Cao (2008) | A survey study of critical success factors in agile software projects | Journal of Systems and Software | Quantitative | None | Project Performance |
| Gupta et al. (2019) | Relationships between IT department culture and agile software development practices: An empirical investigation | International Journal of Information Management | Quantitative | Competing values model | Adoption |
| Campanelli and Parreiras (2015) | Agile methods tailoring - A systematic literature review | Journal of Systems and Software | Literature Review | n/a | Adoption, Tailoring |
| Chuang et al.  (2014) | Assessment of institutions, scholars, and contributions on agile software development (2001-2012) | Journal of Systems and Software | Literature Review | n/a | Review |

### Themes in the Agile Literature

Strode (2016) reported that agile software development research had a broad scope, including information systems, IT project management, and software engineering. As can be seen in Table 6, within the literature reviewed for this study there were several recurring themes within these categories. Namely, IT-enabled agility, project performance, agile usage, and agile adoption. This section will present a summary of existing research regarding IT-enabled agility, project performance, and agile usage in order to provide context before focusing in more depth on the agile adoption literature providing the foundation for this study.

#### IT-enabled agility

Within the basket of eight literature, a number of papers (7 out of 18) focused on how IT might enable agility (Chakravarty et al., 2013; Lee et al., 2015; Lowry & Wilson, 2016; Lu & Ramamurthy, 2011; Nazir & Pinsonneault, 2012; Sambamurthy et al., 2003; Tallon & Pinsonneault, 2011).

Chakravarty et al. (2013) suggest that IT competencies serve two purposes, one facilitating and one enabling agility. They place particular importance on the role of IT competencies in “*activating the firm’s agile capabilities and enhancing performance”* however, they remark that although possession of increased IT competencies may enable increased agility, that in itself is not enough. It is the interaction between the organisation and IT competencies which in their view will lead to increased agility*,* although increased environmental volatility added complexity to this relationship. Chakravarty et al. (2013) recognised that as environmental dynamism increases, the enabling effects of IT competencies decreases. They reported that this was due to IT competencies often relying on historical precedence and being too rigid, arguing that *“to maintain agility, IT competencies should continuously evolve”.* Given that agility was borne from a need to respond to fast-paced change and increase flexibility, it is unsurprising to find that facilitators or enablers to agility must also be receptive to change.

Lee et al. (2015) focused on IT ambidexterity, defined as the ability to exploit current and explore new IT resources and practices simultaneously. As with Chakravarty et al. (2013), Lee et al. (2015) found environmental dynamism to be a mediating factor concerning agility. In contrast to Chakravarty et al. (2013), Lee et al. (2015) found that IT ambidexterity was crucial where environmental dynamism was high, as it allowed for simultaneous innovation and improvement of IT capabilities. They argued that much of the literature studying IT capabilities failed to account for such conditional factors and concluded that practitioners needed to carefully balance IT exploration and exploitation in order to maximise their agile capabilities. This was a theme in much of the IT capability literature, with several scholars referring to utilising IT in order to build sensing and responding capabilities within organisations (Chakravarty et al., 2013; Nazir & Pinsonneault, 2012; Sambamurthy et al., 2003).

In summary, the literature suggests that IT capabilities enhance agility in several ways. First, IT capabilities enable agility through the provision of tools to sense and respond to environmental changes, enabling organisations to exploit opportunities. Second, IT ambidexterity facilitates agility by allowing for simultaneous exploration and exploitation of IT resources and practices, facilitating learning and innovation, which are crucial to an agile approach.

#### Project Performance

Although they focus on different variables, Strode (2016), Chow and Cao (2008), and Lee and Xia (2010) all concentrate on the impact of agile on project performance. Project performance was defined across these papers in terms of quality, scope, time, and cost (Chow & Cao, 2008), with Lee and Xia (2010) suggesting projects needed to be on-time, on-budget and in-scope. However, these metrics align more closely with traditional project management, where deviations from the plan are seen as failings. In contrast, part of the philosophy of an agile approach is to focus on value and customer satisfaction, being responsive to change rather than rigidly adhering to pre-defined controls.

Lee and Xia (2010) found that team autonomy and response efficiency contributed to increased project performance, whereas Strode (2016) focused on handling project dependencies, which they considered to be essential for successful project delivery. Strode (2016) assessed how different categories of agile practices contributed towards dealing with project dependencies, and consequently offered advice regarding which practices organisations should adopt first. They recommended prioritising those practices which addressed multiple project dependencies simultaneously, such as cross-team talk, face-to-face communication, and sprints of 1-2 weeks. In contrast, Chow and Cao (2008) focused on identifying success factors in agile projects, which they categorised under organisational, people, process, technical, and project. They found that choosing a high calibre team, practising agile engineering techniques, and following an agile delivery strategy were the most influential in ensuring project success.

In summary, there were a wide range of factors found to be influential to project success, with several overlapping themes between the papers reviewed. For example, both Lee and Xia (2010) and Chow and Cao (2008) found team level characteristics to be important, such as team selection and autonomy. There were also specific practices highlighted by both Strode (2016) and Chow and Cao (2008) which contributed to project success, such as short sprints and an iterative agile delivery strategy. Interestingly, the measures used to assess project success across the literature were based on traditional project management thinking focusing on cost, scope, time, and quality. This seems to conflict with agile principles, which place customer satisfaction, delivery of value, and embracing change as core characteristics.

#### Agile Usage

Cho (2008) and Beynon-Davies et al. (2000) carried out case studies on the usage and application of agile methods. Cho (2008) focused on Scrum, whereas Beynon-Davies et al. (2000) focused on rapid application development (RAD). Beynon-Davies et al. (2000) argued that descriptions of actual development work and an interpretive perspective were rare in the IS literature. They found that the difficulties in adopting RAD included the cost involved, how it would scale, the justification for changing approach, and the necessary cultural changes. These are issues commonly identified in the literature regarding method adoption and are not exclusive to RAD. However, Beynon-Davies et al. (2000) did not address how these issues might be dealt with, a concern this study seeks to inform.

Cho (2008) identified several challenges to the usage of Scrum. These included knowledge sharing (because of a lack of documentation), issues with communication between autonomous teams, difficulties engaging busy users, conflicting views of developers regarding effective working environments, and the perception that some scrum ceremonies, such as the sprint review, were wasteful. Arguably these could be considered cultural changes using the categories (cost, scalability, justification, culture) defined by Beynon-Davies et al. (2000). Cho (2008) suggest that to maximise the success of projects using scrum, organisations should address each of these five challenges and resolve them before launching the project.

As with Beynon-Davies et al. (2000), Cho (2008) did not offer advice on how the challenges identified should be resolved. They only state that they should be addressed before the project starts. A lack of advice regarding resolving these challenges leaves a significant gap for practitioners. It also appears to conflict somewhat with the agile philosophy, which encourages continuous improvement and learning through doing.

In summary, the agile usage literature provides insight into the day-to-day use of a selection of agile methods. Through this, challenges such as cost and cultural change have been identified. However, there remains a lack of advice for practitioners on how these issues should be resolved. This is a gap this research seeks to address through an in-depth exploration of factors influencing adoption at SoftwareCorps, and the analysis of how these manifest in day-to-day software development activities.

### Adoption of Agile Working Practices

The focus of this research is on understanding how agile principles and practices are adopted in software development teams, as such an appreciation for existing research on the adoption of agile and the adoption of working practices is essential.

In their 2002 study, Riemenschneider et al. (2002) identified a lack of research focusing on methodology adoption in software engineering and urged further research on the topic, however research in this area is still relatively limited. Consequently, to provide a view of current knowledge on methodology adoption, studies from the manufacturing industry were also explored for this section. Given lean and agile have roots in the manufacturing industry, most notably from within the Toyota Motor Corporation, this seemed fitting. Papers that focused on the adoption of working practices in general, rather than specifically agile working practices, have also been included for this section. Table 7 below provides a summary of the adoption specific literature included in Table 6, with the addition of papers from the manufacturing and working practices domains.

Table 7: Adoption Factors and Approach

| **Author** | **Adoption Factors** | **Approach to Adoption** |
| --- | --- | --- |
| Cram and Newell (2016) | Whether adoption exhibits characteristics of mindfulness (e.g. reluctance to simplify complex situations, vigilance in attending to small operational details) or mindlessness (e.g. lack of attention to specifics, contentedness to be a follower) | Three approaches to adoption: Crusaders, who exclusively employ agile in a pure form; Tailors, who customize agile to fit their specific circumstances, while concurrently employing traditional techniques; and Dabblers who employ a few ceremonial agile techniques on top of a fundamentally traditional development approach |
| Vidgen and Wang (2009) | Time pacing, self-management with discipline and routinization of exploration are among the agile enablers | n/a |
| Hovorka and Larsen (2006) | Network organization characteristics and communication processes that reinforced social influence and supported knowledge transfer positively influenced adoption agility | Creation of a strong, dense network of communication ties that support the social information processing of knowledge, attitudes, and behaviours that influence organizational adoption decisions |
| Fitzgerald et al. (2006) | n/a | An a la carte selection of practices and tailoring of practices can work very well. |
| Strode (2016) | n/a | Focus first on practices which address multiple project dependencies |
| Gupta et al. (2019) | IT department culture | Before organizations deploy agile software development practices, they need to understand the existing cultural profile of their IT departments and the desired cultural profile required to build the readiness that is critical for the success of agile adoption |
| Campanelli and Parreiras (2015) | Organization culture  Resistance to change  Need for upper management sponsorship and involvement | n/a |
| Black and Lynch (2001) | It is how the work practice is adopted that matters more than the practices themselves | Joint decision making coupled with incentive-based compensation have higher productivity than businesses that maintain more traditional labour management relations |
| Pil and MacDuffie (1996) | Level of complementary organisational practices and technologies that would increase the benefit from the new practices  Performance levels the organisation is achieving with its current practices  Organisational characteristics or actions that alter the cost of introducing the new practices | Take a systems perspective on adoption – complementarities between practices.  Allow adequate time for practices to develop.  Ensure complementary HR practices compliment high involvement working practices |
| Riemenschneider et al. (2002) | Organizational mandate to use the method  Compatibility of method with existing practice  Buy-in to using the method | Adapting new working practices to more closely match existing practices  iteratively improving them so as not to change practices in a “big bang” approach |
| Tolfo and Wazlawick (2008) | Fundamental cultural changes – open to new technologies and ways to work  Satisfy technicalities of a new practice | n/a |
| Chau and Tam (1997) | Perceived barriers to adoption  Expertise/competence in what is being adopted  Compatibility with vested interests of employees | Reduce knowledge barriers |
| Staron (2006) | High initial investment  Unclear benefits of practices | Staged adoption, starting small and growing usage  Steep learning curve and initial failures |
| Ichniowski et al. (1995) | Cost of switching work practices  Resistance to change  Employees vested interest in old ways of working  Trust between management and employees | Adopt clusters of complementary work practice.  Combinations [of practices] that emphasize high levels of employee participation in work teams, flexible job design, heavy reliance on incentive pay, and extensive training and communication produce significant productivity advantages |
| Ramirez et al. (2007) | Alteration of power relations – vested interests  Presence of a threat making adoption beneficial | Engage all effected parties in decision and development of working practices |
| Kostova and Roth (2002) | Existing knowledge and expertise of practices  Identification and Trust  Social knowledge about a practice (shared cognitive categories)  Values and beliefs consistent with practice | Mimetic and normative rather than coercive adoption |

#### Factors influencing adoption of working practices

As a result of the variability in the adoption of agile practices and the complexity of the adoption process, several scholars explore factors that influence agile adoption. Tallon et al. (2019) categorised these into four groups; technological, behavioural, organizational/structural, and environmental. Across the working practices adoption literature found for this review, there were a variety of factors identified that influenced adoption.

For example, Riemenschneider et al. (2002) found that the presence of a mandate to use a method, the compatibility of that method with existing practices, and the opinions of users towards the method as influential in the successful adoption of a methodology by software developers. Pil & MacDuffie (1996) tested a number of hypotheses regarding the adoption of work practices in the manufacturing industry and concluded that there were three drivers impacting adoption;

“(1) the level of complementary organisational practices and technologies that would increase the benefit from the new practices, (2) the performance levels the organisation is achieving with its current practices, and (3) organisational characteristics or actions that alter the cost of introducing the new practices.” (Pil & MacDuffie, 1996)

Several scholars identified compatibility or similarity of new working practices with old practices as an enabling factor to adoption, including Riemenschneider et al. (2002) and Fitzgerald (2006). It is unsurprising to find this is the case given barriers to change are lower the more familiar the practices are. However, given agile working practices are a significant shift away from traditional methods, this is somewhat unrealistic to expect in practice. If research suggests innovation of working practices is most successful when constrained by how organisations have worked in the past, this does not lend itself to an agile philosophy of experimentation and learning. It does perhaps go some way to explaining why some organisations struggle with the transition.

The importance of organisational culture on the adoption of new working practices was a theme in several studies reviewed (Campanelli & Parreiras, 2015; Gupta et al., 2019; Pil & MacDuffie, 1996; Tolfo & Wazlawick, 2008). Strode (2016) cited Iivari and Iivari (2011) to argue that issues with adopting agile methods in different organisational cultures are still not well understood. Tolfo & Wazlawick (2008) argued that fundamental cultural changes had to take place in addition to satisfying the technicalities of a new practice, and they saw the necessity of changing values and attitudes as the biggest challenge to this kind of change. This echoes comments made by Gupta (2019), who suggest that before organizations deploy agile software development practices, they need to understand the existing cultural profile as well as the desired cultural profile required for successful adoption. It was evident from the literature that the need for cultural adaptation added significant complexity to the adoption of agile working practices (Campanelli & Parreiras, 2015) and that this needed consideration from the outset of an agile transition. An awareness of this theme informed the analysis of this study. As is noted by LeCompte and Goetz (1982), ‘culture' as a concept is often defined differently by different researchers, from a global definition of *a way of life of a people*, to a more specific observation of certain behaviours, or even denial that it exists independently as a construct. In this study, rather than adopting a high-level view of culture, lower-level facets seen to shape culture, such as leadership, became the focus.

Kostova and Roth (2002) suggested that taking a mimetic (copying successful patterns) or normative (adopting patterns considered appropriate) approach, rather than a coercive (patterns imposed by a more powerful authority) approach to the adoption of agile working practices would lead to more successful adoption. In a similar vein, Black & Lynch (2001) concluded that *how* new workplace practices are implemented has more consequence that *what* is implemented;

“Adopting a TQM system per se does not raise productivity. Rather, allowing greater employee voice in decision making is what seems to matter most for productivity” (Black & Lynch, 2001).

In light of this, Black & Lynch (2001) suggested mitigating actions that could be taken to maximise the chance of success. These included adapting the new working practices to more closely match existing practices, and iteratively improving them so as not to change practices in a "big bang" approach (Riemenschneider et al., 2002). Fitzgerald et al. (2006) also supported the idea of selecting practices that fit more closely with the existing ways of working, adding that their findings refuted the suggestion that agile methods were only successful when practices were used together to create synergies. In contrast, Pil & MacDuffie (1996) discouraged such an approach given what they call "complementarities", which come from the combination of different practices. They called for a systems approach arguing that the performance of new work practices, when used together with "complementary" practices, is far higher than the performance of any practice carried out alone.

Additionally, Pil and MacDuffie (1996) found that when installed together with new practices, complementarities allowed for greater success and adoption overall. Ichniowski et al. (1995) agreed and suggested complementarities were to be found through the clustering of work practices. Strode (2016) noted that early adopters of agile believed that agile practices must be adopted in their entirety as they supported each other. However, as agile methods increased in popularity, organisations were seen to mix and match practices from different methods to suit their needs. Often this was done without understanding the reasons practices from a given method needed to be combined (Strode, 2016). Strode (2016) also observed that deciding which agile practices to adopt was a known problem in the adoption of agile methods. It appears, therefore, that there are conflicting opinions regarding the necessity and preference for the adoption of 'whole' methods versus tailoring and taking an 'a la carte' approach, and a lack of research to provide empirical evidence on the subject. This study adds to the debate by exploring how agile practices interact and relate to each other in a real-world context.

Pil & MacDuffie (1996) ground their research in the evolutionary economics and innovation literature, and highlight the reality of 'organisational routines' which are ingrained in the culture of an organisation and seldom change. They also distinguish between minor 'evolutionary' changes and fundamental 'revolutionary' change, which are characterised in some cases as 'competence-destroying' (Pil & MacDuffie, 1996). For example, the introduction of a team centred model of work to a traditionally hierarchical organisation was presented as competence-destroying by the authors. In this case, individuals whose roles were defined by the hierarchical model (e.g. project managers, middle managers, supervisors) found that their existing skills were no longer valued. This was an issue also raised by McHugh et al. (2011), who noted that devolving substantially more control to the team presented a dramatic change for the traditional project manager, who had previously had primary control of the team.

Linked to this were comments made by several scholars regarding resistance to change based on the vested interests of individuals (Chau & Tam, 1997; Ichniowski et al., 1995; Ramirez et al., 2007). As with Pil and MacDuffie (1996), Chau and Tam (1997) found that where an individual's skills would become less relevant, or even obsolete as a result of the change, they would naturally be resistant to that change. Ramirez et al. (2007) found that when those affected by the change were not part of the design and definition of it, they felt their interests had been ignored, and as such, they were less likely to cooperate. This suggests that the buy-in and support of those in leadership positions are essential to agile adoption, as will be explored as part of this study. Without it, those whose roles are defined by incumbent structures present barriers to the transition. Pil and MacDuffie (1996) discussed how this created a barrier between old and new ways of working, which could be made harder to overcome if the cost of change was perceived as too high in the short term, obscuring potential longer-term benefits;

"High-involvement work practices may represent "competence-destroying" change, which is challenging to implement and may lead to a worsened performance in the short term (and this not an economically rational choice for individual managers held accountable for short term results)." (Pil & MacDuffie, 1996)

According to Pil & MacDuffie (1996), new working practices need time to develop before they deliver improvements for the organisation. However, they must first overcome 'competence-destroying' up-front costs. Without a means to address this cost and considering the time such changes require to become established, businesses may struggle to realise success in changing their working practices organisationally. Similarly to the comments of Pil & MacDuffie (1996) on the “revolutionary” change required to adopt new working practices, Riemenschneider et al. (2002) remark on the “radical” change required to adopt new methodologies and the associated difficulties with adoption. They again highlight the upfront cost on productivity and the need for individuals to gain knowledge of the new way of working, which in turn impacts their ability to be productive.

It was a common theme throughout the literature that existing organisational processes and policies needed to complement any new working practices (Black & Lynch, 2001; Pil & MacDuffie, 1996; Riemenschneider et al., 2002; Tolfo & Wazlawick, 2008). There was agreement that if existing policies remained in place which conflicted with the new work practices, there was a high risk that they would work against successful adoption. Reward and remuneration was consequently a topic covered during interviews for this study, although it was not a dominant theme. For example, the literature suggests that an individual who is required to learn an entirely new way of working, while continuing to be rewarded based on individual productivity, is less likely to adopt the new practices (Riemenschneider et al., 2002). In the words of Riemenschneider et al. (2002), *“Positive consequences for the organization cannot offset negative consequences for the individual”.*

Strode (2016) suggested that variation in how agile software development is carried out occurs because adoption is based on prior experience, training from external training organisations, self-teaching, or hiring consultants to coach a team. They argue that there is a lack of research-based guidance regarding which practices to adopt, why to adopt them, and how they are effective. Campanelli and Parreiras (2015) reflected that despite both scientific and market evidence that agile methods are effective, the complexity of adoption is high due to organisational culture, resistance to change, and a need for management support as well as significant efforts from both the organisation and teams. These themes are apparent in a number of the studies included in this review, as discussed in this section. Cram and Newell (2016) posit that although past research has investigated the tailoring of agile practices, it is still unclear what factors are driving adoption variability and how the use of agile techniques differ. Adding to our understanding of this variability is central to this study.

#### Behavioural vs. cognitive adoption

The concepts of behavioural and cognitive change are well-established in the organisational learning (OL) domain. For example, Leroy and Ramanantsoa (1997) and Tsang (1997) noted that a distinction is often made in the OL literature between behavioural and cognitive organisational learning, and that cognitive change is required for ‘real’ organisational learning to occur (Leroy & Ramanantsoa, 1997). Similar concepts appear in the adoption literature, although different terms are used for this in the agile domain. Cram and Newell (2016) spoke of *mindful* versus *mindless* adoption and Kostova and Roth (2002) of *ceremonial* rather than *genuine* adoption. Despite using different terms, both Cram and Newell (2016) and Kostova and Roth (2002) identified a distinction between these two types of change.

For example, Kostova and Roth (2002) define ceremonial adoption as the formal adoption of a practice without the belief there is real value in doing it. As such, the practice is not internalised (essential for adoption), and positive attitudes towards it are not formed. They suggest this type of adoption occurs when there is a lack of belief in a practice combined with strong pressure to adopt it. They emphasise the need for values and beliefs to be aligned with practices in order for adoption to be more than ceremonial – to achieve genuine adoption. The term ‘ceremonial’ could comfortably be applied in this study given ‘ceremony’ is already used in an agile context to depict events such as the daily stand-up, sprint planning, and retrospective. However, the term ‘genuine’ may be more divisive. To claim something is ‘genuinely’ agile suggests a final state at which agile can be achieved. This conflicts with agile concepts such as regularly reflecting to improve and embracing change, where teams are encouraged to continuously evolve how they work as they learn more. It also suggests that anything other than this state is not genuine. The term ‘genuine’ does not fit with the view this study adopts where agile is seen as a spectrum, as will be presented in Chapter 8. The terms *ceremonial* and *genuine* are therefore not adopted for this study.

Cram and Newell (2016) used the terms *mindful* adoption, where individuals are conscious of complexity and vigilantly attend to details, versus *mindless* adoption where individuals exhibit a lack of attention to specifics and are content to follow where they are led. Where adoption exhibits mindfulness, they consider this revolutionary. Where adoption exhibits mindlessness, they consider this a fad or trend – something that will go in and out of fashion. They drew on management fashion theory when choosing these terms. Aspects of this terminology fit well with this study, such as the concept of mindlessly being led. However, the perspective this study takes on the adoption of agile focuses on the values and beliefs of those adopting an agile approach, and how this influences the type of change achieved. Management fashion theory and its associated terms did not appear to convey this and, as such, the terms mindful and mindless were not adopted for this study.

Consequently, rather than drawing on concepts from within the agile literature, such as those given by Cram and Newell (2016) and Kostova and Roth (2002), terms for this study (cognitive and behavioural) were taken from the organisational learning (OL) domain. In this context, cognitive change is concerned with knowledge, understanding and insights (Tsang, 1997). Cognitive change redefines values, beliefs and world views of individuals and organisations (Fiol & Lyles, 1985). In contrast, behavioural change is focused on actions (Tsang, 1997) and the adjustment of parameters (Fiol & Lyles, 1985).

Fiol and Lyles (1985) described how *“changes in behavior may occur without any cognitive association development; similarly, knowledge may be gained without any accompanying change in behavior”*. Applying this to agile adoption, it may be possible to see behaviours adopted without individuals developing an understanding of the purpose of agile practices and principles, or for an understanding of agile practice and principles to exist without the ability to change behaviours to align with this. Fiol and Lyles (1985) presented a model of learning and change on which there were four positions, as can be seen in Table 8.

Table 8: Model of Learning and Change adapted from Fiol and Lyles (1985)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cognitive Development | | |
| Behavioural Development |  | Low | High |
| Low | A | C |
| High | B | D |

Fiol and Lyles (1985) identified a need for organisations to determine which position would be most suitable for their environment and strategy. They suggested that position A is typical of many bureaucratic organisations and may be suitable in a stable and predictable environment in which change and learning are not required. They describe Position B as an organisation which continuously takes actions and embarks on changes but does not base this on learning what works. As such there is repeated shock to the organisation without a sense of direction. Position C is described as most appropriate in a turbulent environment where learning is essential for survival but too much change would cause a loss of direction.

Of particular relevance for this study was position D, indicating a high propensity for change and learning. Organisations in this position achieve high levels of cognitive development along with high levels of behavioural change. Fiol and Lyles (1985) suggest that this position is most appropriate for organisations within turbulent environments, where constant change is the norm and focus is on problem solving. Given the parallels this description has with the environment from which agile methods evolved, this suggests a need to take both a cognitive and behavioural approach to agile adoption. Findings from this study support the concept of a distinction between these types of adoption. As such, the terms *behavioural* and *cognitive* are used to describe the types of adoption observed. This builds on existing research, applying concepts from other domains, and adds to the body of knowledge in this area.

### What theory has been used to study agile

Several scholars highlighted a lack of theoretical underpinning in agile software development research (Chuang et al., 2014; Cram & Newell, 2016; Dingsøyr et al., 2012; Lang et al., 2013; Strode, 2016), which they argued was a common criticism in the domain. Cram and Newell (2016) suggested that the number of theoretical perspectives was growing as a consequence of this criticism, with knowledge management and personality-related foundations being prevalent. Dingsøyr and Lassenius (2016), noted that there had been an increase in the number of agile studies connecting empirical findings to theories taken from more mature research fields, such as design thinking from architecture and strategic management domains (Dingsøyr et al., 2012). The following section reviews the theory that has been used to study agile software development methods.

Cram and Newell (2016) used the theories of management fashions and mindful innovation to explore how organisations adopt and use agile. They explored how different types of adoption exhibited similarities to trends or fads, as well as which types of adoption were more sustainable. They found that there were three categories of agile adoption; crusaders, tailors, and dabblers. They posited that dabblers were the least likely of the three categories to adopt agile long term, as they were more likely to transition to the next emergent development trend. These concepts were used in this study as a lens to explore different phases of agile adoption at SoftwareCorps. For example, similarities were evident between Cram and Newell’s concept of crusaders and those at SoftwareCorps who sought to adopt agile ‘by the book’.

Hovorka and Larsen (2006) noted that adoption studies often use the diffusion of innovation theory and the technology acceptance model (TAM) as lenses for research. For example, Riemenschneider et al. (2002) used 5 theoretical models traditionally used to assess tool adoption to gain insight into the intentions of developers to adopt methodologies. These included TAM, TAM2, PCI, TPB, and MPCU. Despite acknowledging clear differences between the adoption of tools and the adoption of methodologies, Riemenschneider et al. (2002) also noted key similarities. Namely that both require behavioural change and effort to learn. This study makes a distinction between the tools and practices of a method and the underlying philosophies and principles which guide it. As such, theories traditionally used to assess tool adoption were discounted, given they did not allow for the inclusion of the 'softer' aspects of agile adoption.

Of the papers selected for this review of the adoption literature, three theories were commonly used or discussed; Capability building (Lee et al., 2015; Sambamurthy et al., 2003), contingency theory (Chakravarty et al., 2013; Fitzgerald et al., 2006; Lowry & Wilson, 2016) and complex adaptive systems (CAS) theory (Alaa & Fitzgerald, 2013; Meso & Jain, 2006; Vidgen & Wang, 2009). Of these, CAS theory has been advocated for most strongly by the agile community (Vidgen & Wang, 2009). A complex adaptive system (CAS) is capable of adapting to its external environment and internal state such that it survives despite its circumstances (Meso & Jain, 2006). The holistic system is said to be more than the sum of its individual parts (Meso & Jain, 2006). These parts are autonomous and have the ability to intervene meaningfully in the course of events (Vidgen & Wang, 2009). A CAS is typically unpredictable whilst exhibiting various forms of order and regulation (Alaa & Fitzgerald, 2013). Vidgen and Wang (2009) cite Highsmith (2002, p. 48), arguing that CAS is not only an appropriate theory for software development but *“the only way to make sense of the world”.* Early in this study CAS was considered as a lens through which to explore the research questions given its prevalence in the agile literature. However, as focus narrowed on leadership and agile adoption, transformational-transactional leadership theory was found to provide more insight on the case.

### What research methods are used most?

Chuang et al. (2014) found that the most widely used research methods in agile software development studies were the case study, design simulation, and experiments - methods they noted are typical in empirical research. In their review of the literature, Tallon et al. (2019) found a broader cross-section of methods dividing studies into conceptual, empirical - case data, and empirical - survey data. Of the papers found for this review (see Table 6), the split between qualitative (11) and quantitative (13) studies was almost even, with a significant number of papers focused on reviewing the literature (5). Only a small number of papers took a conceptual approach to theory building (2) or used mixed methods (3).

Suggesting further research based on their study of agile as a management fad, Cram and Newell (2016) suggested that benefits could be gained by conducting longitudinal research to examine the extent that an organization may evolve in its agile adoption. They noted that this could be particularly useful in companies that had adopted, abandoned, and re-adopted an agile approach, as is the case at SoftwareCorps. Chau and Tam (1997) and Kostova and Roth (2002) also both called for more longitudinal research in order to better understand complex organisational phenomena. This study, which was carried out over three and a half years, addresses this call.

It was also noted that in many cases, executive management or project management level representatives were targeted for surveys and interviews, meaning that a large proportion of the organisation was not represented in the findings. A large part of the agile philosophy involves placing the responsibility and capability for work and decisions in the hands of those doing the job. As such, it is vital to have a cross-sectional representation of the organisation when designing research in this area. This was a careful consideration in the design of this study, which involved interviews at all levels of the hierarchy, including C-level, middle managers, and importantly software development team members with varying levels of seniority (junior to senior).

## Summary

This chapter has provided the definition of a methodology for this study, positioning the use of terms such as philosophies, principles, and practices of agile within the broader concept of agile as a software development methodology. It has also defined what is meant by agile; the ability to thrive and prosper in an environment of constant and unpredictable change, not only to accommodate change but to also relish the opportunities inherent within a turbulent environment (Maskell, 2001).

It has provided a brief history of agile software development, its evolution, and the problems it seeks to address. From its origins within the lean manufacturing industry to a revolution of software engineering. Shifting away from an ineffective, bureaucratic and controlling system to a more humanistic approach, increasing flexibility and agility, and facilitating learning and continuous improvement. The publication of the Agile Manifesto in 2001 formalised this shift, with interest and research in agile software development continuing to increase. This chapter has also contextualised this study in terms of the existing body of knowledge regarding agile software development, adoption, and working practices.

The literature suggested that IT capabilities enhance through the provision of tools to sense and respond to environmental changes, enabling organisations to exploit opportunities. IT ambidexterity also facilitates agility by allowing for simultaneous exploration and exploitation of IT resources and practices, facilitating learning and innovation, which are reportedly crucial to an agile approach.

A wide range of factors were found to be influential to project success in the literature; team level characteristics were found to be important, such as team selection and autonomy. Specific practices for project success were highlighted, such as short sprints and an iterative agile delivery strategy. Interestingly, measures used to assess project success focused on cost, scope, time, and quality, which seems to conflict with agile, which places customer satisfaction, delivery of value, and embracing change as core to success.

As evidenced in the literature, despite an increase in the popularity of research on agile software development, there are still some significant gaps. Riemenschneider et al. (2002) identified a lack of research focusing on methodology adoption in software engineering, and both Dikert et al. (2016) and Strode (2016) highlighted a lack of academic research on agile practice usage. As a consequence, scholars were seen to rely heavily on practitioner reports collated by Version One. This study fills this gap through an in-depth analysis of the day-to-day working practices carried out by software development teams, as well as through the mapping of practices to agile principles to better understand how the adoption of agile practices achieve agility. Understanding this also contributes towards concerns raised by Chakravarty et al. (2013), who stated that we are yet to fully comprehend how organisations build and leverage agility, and those by Sidky et al. (2007) who called for more research guiding the adoption of agile practices. The agile usage literature provides insight into the day-to-day use of a selection of agile methods, calling out challenges such as cost and cultural change to successful adoption and usage. However, there remains a lack of advice for practitioners on how these issues should be resolved.

Despite evidence that agile methods are effective, the literature suggests complexity of adoption is high due to organisational culture, resistance for change, and a need for management support as well as significant efforts from both the organisation and teams. Cram and Newell (2016) posit that although past research has investigated the tailoring of agile practices, it is still unclear what factors are driving adoption variability and how the use of agile techniques differ. Adding to our understanding of this variability is central to this study.

The values and principles of the Agile Manifesto were designed to shape the type of environments in which its creators desired to work, however, it does not go as far as prescribing specific practices. Although principles are universal they are not always easy to apply, practices give specific guidance on what to do. As such agile practices should realise agile principles as demonstrated in the mapping provided in Table 5. A lack of guidance on how agile principles should be manifest leaves much down to the interpretation of those involved in the transition to agile working. This review of the literature suggests this lack of guidance may explain why adoption levels are so variable between organisations and teams. A distinction was observed in the literature between technically- and socially-orientated practices, which shared similarities with the cognitive and behavioural change literature. This distinction suggested a need to consider both a cognitive and behavioural view of agile adoption, and consequently the terms *behavioural* and *cognitive* were adopted for this study.

Leadership is the subject of much research both in academia and in industry, and it is known that leaders set the tone for the rest of the organisation (Hill et al., 2012). As will be described in section 5.2, at SoftwareCorps there were a number of individuals who were charged with leading the software development department at different periods in time, with participants’ noting that *“with each CTO comes a new way of working”* (Quinn, Dev). Although most leaders called their way of working agile, there were clear differences between the working practices carried out and the styles of leadership employed. This will be explored in-depth in Chapter 7. In the following chapter, an overview of the leadership literature will be provided, exploring leadership in both the information systems (IS) and organisational behaviour (OB) domains.

# Literature Review - Leadership

Based on the findings which will be presented in Chapter 6, leadership was identified as a key theme in relation to the adoption of agile working practices at SoftwareCorps. As such an additional phase of data collection was carried out focusing on the impact of leadership on the adoption of agile. This section presents a short review of the key leadership literature.

This section introduces the topic of leadership as discussed in the literature and positions this research in relation to what has already been explored. First, the definition of leadership is explored and a definition for this study is presented. Second, perspectives from which leadership has been considered will be discussed and the perspective taken in this study established. Third, theories of leadership will be presented and justification for the choice of theory for this research given.

## Defining Leadership

Leadership is a domain which has been the subject of much study, both in academia and industry (Karahanna & Watson, 2006). However, definitions of leadership are varied (Barling, 2014; Blom & Alvesson, 2015; Friedman, 2016; Karahanna & Watson, 2006; Ola, 2017; Pauleen, 2003; Yukl, 2012). On reviewing the literature, Karahanna and Watson (2006) argued that the term is often used imprecisely and posited that *“there are possibly as many definitions of leadership as individuals who tried to define it”.* Given the broad usage of the term ‘leadership’, it is surprising that a significant number of papers tackling the subject don’t appear to define the term at all. Rost (1991, p. 7), for example, found that 60% of authors had not defined what they meant by leadership. It is the purpose of this section therefore to define what is meant by leadership in the context of this study, drawing from the broader literature.

Definitions range dramatically from leadership as psychotherapy or support, to a strong hierarchical subordinate relationship (Blom & Alvesson, 2015). Pauleen (2003) highlighted this, stating that leadership can be viewed *“as a structured authoritative role, or as the ability of individuals to intrinsically or extrinsically motivate followers”.* The motivational aspect of leadership was noted by other scholars, such as Charlier et al. (2016) who quote Yukl (1989) to define leadership as follows;

“Leadership, broadly defined, is an “influence process involving determination of the group's or organization's objectives, motivating task behavior in pursuit of these objectives, and influencing group maintenance and culture”.“

It appears, therefore, that the ability to influence and motivate others are central facets of a leadership definition. A link between leadership and motivation and purpose can also be seen in the industry literature. For example, Kerr (2013, p. 35) argues that people identify with things that resonate with their personal values and beliefs, and that tapping into this has the ‘emotional power’ to shape behaviour. Kerr (2013) considered leadership as the ability to bring people together, empower them, demonstrate shared beliefs, and create a relationship of mutual trust and understanding. Similarly, Kurtzman (2010, p. 26) stated that;

“People have a need to be heard….Great leaders…grant [people] their space, give them their trust, allow them responsibility, and present them with opportunities and resources to do their jobs” (Kurtzman, 2010, p. 26)

In contrast, Pfeffer (1997, p. 126) focused more on the authoritative aspects of leadership, defining it as *“Control…exercised through individual, interpersonal influence, in which those in roles of authority motivate and direct others to act as they would like”*. Gardner (2007), on the other hand, commented that *“we have all occasionally encountered top persons who couldn’t lead a squad of seven-year-olds to the ice cream counter”* and therefore suggested that leadership is not equal to authority or power;

“the thug who sticks a gun in your ribs has power. Leadership is something else…Leadership requires major expenditure of effort and energy – more than most people care to make”

In a recent review, Ola (2017, p. 57) took a thematic approach to define leadership as belonging to those who can *“transform their concepts, convictions and vision into organisational changes through their influence over other people”*. These views suggest that leadership is not something which can be given through assigning authority through position, such as in making someone a manager. Rather, it is something more characteristic of an individual, requiring a conscious decision to lead.

Blom and Alvesson (2015) argued that, rather than using the term leadership as a nebulous signifier under which anything relating to organising exists, scholars should instead focus on the activities being referred to. They narrowed down key leadership activities from the literature and thematically grouped them under; direction (influencing ideas, values, meanings, and understandings of followers), support (cognitive and emotional) and inspiration (provide a moral example, a role model). They also noted a distinction in the literature between leadership and management, the latter being more to do with formal authority and the use of control and compliance rather than influence. The distinction between leadership and management resonated strongly with this research, including characteristics of support, motivation, and inspiration. As such, definitions of leadership which included these attributes in their breakdown were drawn on to define leadership for this study. This definition is as follows;

Leadership is the ability of an individual to influence others through activities of direction, support, and inspiration. Leaders provide direction through uniting people around a common vision. They support through empowering and trusting those they lead to do what is needed to achieve the shared vision. They inspire through providing followers with purpose, leading by example, and demonstrating shared values and beliefs.

Having defined leadership, the following section will focus on the perspective from which leadership has been studied in the reviewed papers. This will provide the foundation from which the perspective of this study will be presented.

## Perspectives on Leadership

As with the nebulous definitions of leadership, the perspectives from which leadership has been studied are also diverse (Karahanna & Watson, 2006; Ola, 2017). Early research focused on the traits of the leaders themselves with the belief that certain individuals were predisposed to lead – evident in the ‘Great Man’ theory (Cawthon, 1996). Subsequently behavioural and situational leadership perspectives became popular, focusing on the actions of leaders and how their styles changed depending on the context (Ola, 2017). More recently there has been a shift towards leadership traits in general, rather than specifically of an individual. These classify different types of leadership from charismatic (inspiring followers), to transactional (incentivising followers) (Karahanna & Watson, 2006).

Through their review of the literature, Ghawe and Brohman (2016) categorised the available research into two categories. Those that studied leadership *characteristics* (personal aspects of leaders such as emotional intelligence and communicativeness) and those that focused on leadership *styles* (such as transformational and transactional leadership). In both cases behaviours were used to demonstrate the given characteristics or style, for example ‘lives the values’ was representative of a fundamental leadership characteristic.

There have also been studies which looked at the philosophy of leaders and how this impacted characteristics, behaviours and styles. For example, Evans (2007, p. 144) posits that *“authentic leaders build their practices outward from their core commitments rather than inward from a management text”* and that their philosophy, shaped by their life and work experiences, influence everything they do. Overall there appears to be a dichotomy in the literature between those seeking to define the singular ‘best’ form of leadership and those who focus on leadership as it varies in a situational context (Kezar, 2004).

As discussed in this section, perspectives on leadership in the literature vary with studies spanning characteristics (e.g. Rubin et al., 2005) behaviours (e.g. Yukl, 2012), philosophies (e.g. Kezar, 2004), and styles (e.g. Friedman, 2016). Ghawe and Brohman (2016) noted that in comparison to studies on characteristics which were prevalent, leadership style had relatively little attention in the literature and called for more research on what kind of leadership style is optimum for a particular industry sector. This study will seek to answer this call by taking a leadership style perspective to determine whether there is a style of leadership more suited to agile working.

## Leadership in the IS Domain

It is important to distinguish between the general leadership literature and the subject of leadership within the IS domain (Hickman & Akdere, 2017), which is the central concern of this study. For example, Karahanna and Watson(2006) argue that leaders of technical functions have a unique set of problems. Both Karahanna and Watson (2006) and Hickman and Akdere (2017) contend that technical leaders require an understanding not only of technology but also business aspects of the organization in order to be effective. Consequently, Karahanna and Watson (2006) call for researchers to focus specifically on IS leadership to *“enhance our understanding of specific constructs, relationships, processes, structures, and mechanisms that underlie unique IS concerns”.* This study seeks to answer this call by focusing on leadership within the field of agile software development. Table 9 provides a summary of recent literature regarding leadership within the IS domain.

Table 9: Summary of Perspectives on IS Leadership

|  |  |  |  |
| --- | --- | --- | --- |
| **Theme** | **Papers** | **Level of Leader** | **Method** |
| Rollout and adoption of information systems | Cho et al. (2011) | Various levels | Quantitative, Survey |
| Shao et al. (2012) | Senior Leadership | Quantitative, Survey |
| Sharma and Rai (2015) | Executives | Quantitative, Survey |
| Shao et al. (2016) | Top Management | Qualitative, Multi-case Study |
| Team Behaviours | Faraj and Sambamurthy (2006) | Project Leader | Quantitative, Survey |
| Gumusluoglu and Ilsev (2009) | Supervisors | Quantitative, Survey |
| Zhang and Bartol (2010) | Middle Managers | Quantitative, Survey |
| Li et al. (2012) | Project Leader | Quantitative, Survey |
| Guhr et al. (2019) | Various | Quantitative, Survey |
| Dispersed Teams | Joshi et al. (2009) | Team Leader | Quantitative, Survey |
| Graham et al. (2015) | Team Leader | Quantitative, Survey |
| Charlier et al. (2016) | Emergent within team | Quantitative, Experimental Testing |
| Eseryel and Eseryel (2013) | Emergent within team | Qualitative, Grounded Theory |
| Pauleen (2003) | Team Leader | Qualitative, Action Learning |
| Agile software development | Dubinsky and Hazzan (2010) | Change Leader | Qualitative, Case Study |
| Xu and Shen (2015) | Project Leader | Research Proposal |
| Xu and Shen (2016) | Team Leader | Conceptual Paper |
| Xu and Shen (2018) | Within team | Qualitative, Case Study |

Much of the recent IS leadership literature is in relation to the rollout and adoption of information systems (e.g. Cho et al., 2011; Shao et al., 2016, 2012; Sharma & Rai, 2015) the effects of leadership on the behaviour of individuals and teams (e.g. Faraj & Sambamurthy, 2006; Guhr et al., 2019; Gumusluoglu & Ilsev, 2009; Li et al., 2012; Zhang & Bartol, 2010) and leadership within dispersed teams (Charlier et al., 2016; Eseryel & Eseryel, 2013; Graham et al., 2015; Joshi et al., 2009; Pauleen, 2003).

Within the rollout and adoption literature, the majority of papers focused on senior and executive leaders (Shao et al., 2016, 2012; Sharma & Rai, 2015). The central theme was the role of leaders in the successful adoption of information systems. For example, Cho et al. (2011) found that leaders were able to increase the perception of organisational support for IS and made employees feel supported and more confident in IS usage. On the other hand, Shao et al. (2016) found that different styles of leadership suited different phases of the enterprise systems lifecycle, with transformational leadership fitting the adoption phase and transactional leadership best fitting the implementation phase.

Within the literature on team and individual behaviours there was a tendency to focus on project and middle managers (e.g. Faraj & Sambamurthy, 2006; Gumusluoglu & Ilsev, 2009; Li et al., 2012; Zhang & Bartol, 2010). Themes included the impact of leadership on creativity, motivation, and innovation as well on intentions. For example, both Zhang and Bartol (2010) and Gumusluoglu and Ilsev (2009) found that transformational leadership had a positive relationship with employee creativity, whilst Guhr et al. (2019) suggested transformational leadership was essential to influencing employee information security behaviour. Li et al. (2012) found that transformational leadership was effective at intrinsically motivating teams whereas transactional leadership was more effective in extrinsic motivation. Transformational and transactional leadership theory will be defined and discussed in more detail in section 3.4.1.

Within the dispersed team literature, the focus was on team leaders and emergent leaders from within the team, rather than higher or executive management. Themes such as communication, perceptions of leadership, influence in non-co-located teams and team effectiveness were evident. For example, Li et al. (2012) concentrated on intrinsic and extrinsic motivation of individuals within virtual teams and the types of leadership which could influence this motivation. Whereas Charlier et al. (2016) compared the perception of leadership between co-located and dispersed team members, finding that co-located team members had significantly higher ratings of leadership for one another versus their non-co-located counterparts.

Another sub-set of the IS leadership literature are papers which present a review of the literature (Ghawe & Brohman, 2016; Hickman & Akdere, 2017; Karahanna & Watson, 2006). These authors focused on specific topics such as information systems, CIO characteristics, and leadership development.

On reviewing this literature it is apparent that there is limited research on the subject of leadership specifically within agile software development, although it has been considered within the conference setting (e.g. Dubinsky & Hazzan, 2010; Xu & Shen, 2015, 2016, 2018). Xu and Shen (2015) cite this gap as motivation for their research which investigates how different types of leadership can improve agile project performance. They also appear particularly prolific on the subject in recent years, presenting on this subject at a number of conferences (2015, 2016, 2018).

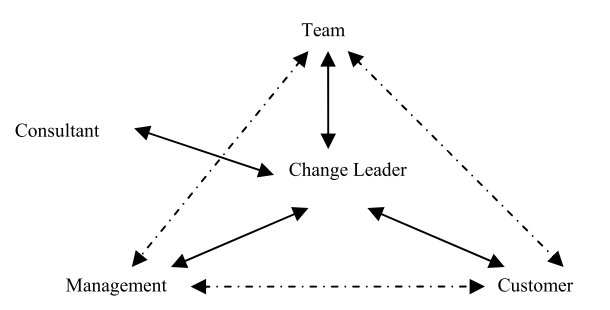
In their earlier research they explored how different types of leadership impacted agile project’s success, focusing on the dynamics between directive, shared, empowering and self-leadership (Xu & Shen, 2015). Later they narrowed their focus to empowering leadership, taking a transactive memory systems (TMS) approach (Xu & Shen, 2016). They found that empowering leadership, along with an established TMS, enabled greater team agility thanks to complementarities in their principles. Empowering leaders encouraged collective actions, developed member capability, resolved conflict, fostered accountability and provided feedback, all of which influenced dimensions of TMS (credibility, specialisation, coordination).

Most recently they adopted a case study approach to build a theoretical model demonstrating the role of leadership in team agility and performance;

“Leadership roles have not been fully investigated from any theoretical lens. In this study, we go beyond the proposed agile practices and examine the role of leadership and how leadership can provide an appropriate environment and support to help achieve agility” (Xu & Shen, 2018)

Although their stated aim is to collect data from multiple cases, their conference paper presented only preliminary findings from a small set of data (6 interviews), from two sites. They focused on the role of the scrum master and product owner, and as such are concerned with leadership within the team and the different roles leaders within an agile team need to play; observer, navigator, initiator, cultivator, external coordinator. Their ultimate objective is to contribute to leadership theory in the context of agile software development as it pertains to team building.

More closely related to this PhD research, Dubinsky and Hazzan (2010) examined leadership as it pertains to transitioning to agile software development. Their focus is on the ‘Change Leader’ (CL), an insider of the organisation who acts as a mediator between parties involved in the transition. They present a model, *ad-hoc* leadership, which they suggested could be used to analyse transition and change processes. As can be seen in Figure 2, the ad-hoc leadership model accounts for the relationships between key parties in an agile transition; the team, the management, and the customer.

Figure 2: Ad-hoc Leadership Model by Dubinsky and Hazzan (2010)

The change leader can be seen to move towards any of the three poles as circumstances dictate. As the frequency of these movements decrease they posit that the ad-hoc leadership style diminishes and may be replaced by alternate styles. Although this model is presented as a model of leadership, it is more concerned with the interactions between the CL and the other parties during an agile transition, and the need for the CL to adopt a different style depending on which pole they are servicing. As such, this model does not inform how leadership at the management (e.g. CTO) level influences the adoption of agile, a central concern of this study which differentiates it from existing research. In presenting this model Dubinsky and Hazzan (2010) add to an already crowded body of theories, as will be discussed in the following section.

The primary research method used to study leadership in the papers reviewed was quantitative, with most selecting survey as the tool (e.g. Cho et al., 2011; Graham et al., 2015; Gumusluoglu & Ilsev, 2009; Joshi et al., 2009; Li et al., 2012; Zhang & Bartol, 2010). The small number of studies reviewed which adopted a qualitative approach varied in their methods. Shao et al. (2012) took a snapshot view carrying out interviews at a single point in time, Xu and Shen (2018) carried out two sets of interviews in 2010 and 2014, and Dubinsky & Hazzan (2010) worked as external consultants with their case organisations for between 6 and 24 months.

Despite research by Dubinsky and Hazzan (2010) and Xu and Shen (2015, 2016, 2018) in the domain of leadership and agile software development, there is still a lack of literature surrounding the influence of leadership on the adoption of agile working practices. This gap will be addressed through in this study which adopts a qualitative case study approach. Given the lack of this type of research in the existing agile adoption literature, this longitudinal research provides a fresh perspective on the phenomena at the CTO level.

In the following section leadership theory will be introduced along with justification of the choice of theory for this study.

## Leadership Theory

According to Gardner et al. (2010) and Dinh et al. (2014), leadership research has had a resurgence in recent years. As a result, there appears to have been a proliferation of leadership theories (W. L. Gardner et al., 2010). In particular an increase in neurological perspectives on leadership, theories relating to leadership and creativity/innovation, the dark side of leadership, and strategic leadership are apparent (Dinh et al., 2014). Dinh et al. (2014) described the state of research regarding leadership as follows;

*“Several established leadership theories continue to capture the interest of the field including neo-charismatic, information processing, trait, and leader-follower exchange theories. However, other leadership theories have not witnessed significant growth, including behavioral approaches, contingency theory, and path-goal theory”* (Dinh et al., 2014)

Both Barling (2014, p. 4) and Dinh et al. (2014) noted that transformational/ charismatic and transactional leadership (categorised under neo-charismatic theories) make up the majority of recent leadership research. Barling (2014, p. 4) highlighted that research on charismatic and transformational leadership was studied more between 1990-2000 than all the other theories put together. Van Knippenberg & Sitkin (2013) credit the interest in charismatic and transformational leadership theory for reinvigorating a field they suggest was lacking excitement in the mid 1980’s.

Dinh et al. (2014) carried out an in-depth qualitative review of leadership theories used between 2000 and 2012, searching the top academic journals based on impact rating and ranking lists in the organisational behaviour domain. They found 66 distinct leadership theories, categorising them as ‘established’ or ‘emerging’. As Barling (2014) found for the preceding decade, Dinh et al. (2014) demonstrated that neo-charismatic theories such as transformational leadership continued to dominate between 2000-2012.

Despite describing the leadership field as *‘increasingly diverse and rich in theoretical insight’,* Dinh et al. (2014) challenged future research to work with the vast body of existing theories to develop integrative perspectives on how they might relate to each other, or work together to result in leadership phenomena. Behrendt et al. (2017) and Van Knippenberg & Sitkin (2013) reiterate this problem and call for more sophisticated and integrative leadership theories that focus on research outside of the core leadership literature. Across the field, leadership research aims to identify the qualities that differentiate successful leaders from their less successful counterparts, and many theories have been established to seek answers to this question (Behrendt et al., 2017).

Dinh et al. (2014) called for a unification of existing theories which they believed would result in new and innovative research for the future. They appealed for a deeper understanding of the dynamics between a leader’s influence on underlying processes and subsequent organisational outcomes, closing with an invitation to scholars;

*“consider how processes change and evolve as they are influenced by context, as well as by leadership occurring from multiple sources within organizations…Linking processes to outcomes can advance theory, and it will also provide a firmer basis for leadership interventions.”* (Dinh et al., 2014)

They also identified a gap in the existing research, suggesting a need for longitudinal study of leadership over time and a greater understanding of *‘the complexity that defines real individual, group, and organizational systems’* (Dinh et al., 2014).

There is a clear call for researchers to expand the study of leadership theory outside of the traditional leadership domain, to understand more about how leaders influence process, and to adopt a longitudinal approach in studying this phenomenon. This study answers these calls by taking a longitudinal perspective of leadership and how it influences the adoption of agile working practices in software development teams. This is particularly relevant in the IS domain where leadership literature is sparse. It was also clear that transactional and transformational leadership theory was not only current, but more importantly an effective lens through which to inform the research aim, as will be discussed next.

The following sections will explore transformational and transactional leadership theory in more detail, providing definition and characterisation which will be used in presenting the phase two findings from this study.

### Transformational and Transactional Leadership Theory

Across the literature reviewed, transformational leadership is loosely defined as empowering and intellectually stimulating individuals to seek a higher order of aspirations. These higher order aspirations are those which focus on team and organisational goals, not just individual desires (Balthazard et al., 2009; Cho et al., 2011; Gumusluoglu & Ilsev, 2009; Shao et al., 2016, 2012). Conversely, transactional leadership is generally described as a leader-follower dynamic with an emphasis on reward and punishment to elicit desired behaviours (Evans, 1996; Li et al., 2012; Shao et al., 2016, 2012). Table 10 presents definitions from across the literature in more detail.

Max Weber is cited as having a profound influence on the study of organisations and leadership (Greenwood & Lawrence, 2005). Of particular relevance to this review is his categorisation of three types of leadership: Traditional, charismatic and rational-legal (Burns, 1978, p. 243). Building on Weber’s work, James McGregor Burns (1978) introduced transformational and transactional leadership in the context of political leadership (Turan & Sny, 1996). Burns considered these two types of leadership as distinct. He believed that individuals could be classified as one or the other based on their propensity to either transact with or transform followers (Lowe et al., 1996).

In the IS literature, transformational leadership is often described in contrast to transactional leadership. As such these two styles are often considered a dichotomy. Building further on the work of Burns, Bass (1985) developed a formal theory of transformational and transactional leadership as part of his full-range leadership theory. In this context, transactional and transformational leadership are not opposites, but rather dimensions of the full leadership spectrum;

“It is important to note that in the view of Bass (1985), transformational and transactional leadership are not two ends of a spectrum but two separate dimensions of leadership, and there exists a high inter-correlation between transformational and transactional leadership” (Shao et al., 2016)

Podsakoff et al. (1990) argued that transformational behaviours *augmented* transactional leadership, increasing trust and respect between leaders and followers resulting in motivation to exceed expectations. Podsakoff et al. (1990) also suggested that effective leaders transform the values, beliefs, and attitudes of followers. This aligns with Dansereau et al. (2013) who commented that transformational leadership is a reciprocal process through which both leaders and followers are transformed for the better.

Table 10: Definitions of Transformational and Transactional Leadership from IS Literature

| **Authors** | **Transactional Leadership** | **Transformational Leadership** |
| --- | --- | --- |
| (Gumusluoglu & Ilsev, 2009) | Allocate and coordinate tasks | Transform followers' personal values and self- concepts, move them to higher level of needs and aspirations |
| (Shao et al., 2012)  (Shao et al., 2016) | Transactional leaders consider how to marginally improve and maintain the quantity and quality of performance, how to substitute one goal for another, how to reduce resistance to particular actions, and how to implement decisions.  Exchange of rewards for effort, attends to details, monitors and controls deviations from rules and standards, coordinates and takes corrective action. | Provide strategic vision, sense of mission, charismatic, instil respect and trust, insightful and knowledgeable, optimistic and enthusiastic  Develop and empower individual through coaching and advising  Raise followers to a greater awareness about the issues of consequence |
| (Li et al., 2012) | Characterized by a leader-follower relationship built on a series of bargains or exchanges.  Focuses on defining clear reward and punishment structures  Provide inducements to obtain desired performance by their subordinates. | Intellectually stimulating or inspiring followers to ignore their self-interests for a collective purpose |
| (Evans, 1996) | Transactional calculative forms of leadership. | Charismatic |
| (Cho et al., 2011) | n/a | Focus on inspiring and revitalizing people to perform better  Inspire the values and ideals of followers and ultimately motivate followers to perform beyond expectations  Leading-by-example, coach or mentor followers |
| (Balthazard et al., 2009) | n/a | Articulate strongly-held beliefs and values, stimulate thinking that fosters innovative solutions to problems, and generate high degrees of follower confidence, trust, and admiration  Provide vision that is different from the status quo, exuding confidence, and being innovative or showing special capabilities |

Of the available literature regarding leadership within the IS domain, consideration of transformational and transactional leadership styles are common, including how their use affects the behaviours, primarily the motivation, of those around them (Balthazard et al., 2009; Cho et al., 2011; Eseryel & Eseryel, 2013; Evans, 1996; Faraj & Sambamurthy, 2006; Graham et al., 2015; Gumusluoglu & Ilsev, 2009; Karahanna & Watson, 2006; Li et al., 2012; Shao et al., 2012; Zhang & Bartol, 2010). Still, this literature is less prevalent in comparison to the studies focusing on leadership characteristics (Ghawe & Brohman, 2016). Eseryel and Eseryel (2013) also argue that although transformational leadership is valued and referred to in the IS literature, it is not often explicitly researched in an IS setting. Few studies set out to understand transformational leadership explicitly. Instead, through the exploration of other factors such as motivation and creativity in teams, transformational leadership emerges.

Joshi et al. (2009) and Graham et al. (2015), for example, focus on leadership within virtual or distributed teams. Joshi et al. (2009) found inspirational leadership, a subfactor of transformational leadership, was highly important in dispersed teams in order to help form a cohesive team identity, something which can be difficult in virtual teams. This is in contrast to Graham et al. (2015) who found that transformational leadership had negative consequences in a virtual team environment, possibly due to the lack of face to face interaction.

Both Zhang & Bartol (2010) and Gumusluoglu & Ilsev (2009) focused on the impact of leadership on creativity finding that transformational leadership had a positive effect on the creativity of individuals and their psychological empowerment, something which Gumusluoglu & Ilsev (2009) suggested also leads to a positive association with organisational innovation.

Shao et al. (2016) and Cho et al. (2011) are more traditional in their focus, looking at the effect of leadership style on the adoption and use of software systems in the workplace. Cho et al. (2011) found that IS success can be enhanced by transformational leadership style whereas Shao et al. (2016) suggest that a combination of transformational and transactional leadership styles work best at different phases in the lifecycle.

However, none of the papers reviewed explored how different styles of leadership impact the adoption of agile working practices, or whether a particular style of leadership is more suited to agile working. This appears to be a gap in the literature and is an area this study will explore.

In order to identify the style of leadership (transformational, transactional or otherwise) it is essential to understand what constitutes transformational and transactional leadership. In the following sections the characteristics of each style will be discussed in order to demonstrate how each style has been recognised in existing research. This then provides a basis from which the subsequent findings can be explored. First, a brief overview of transactional leadership characteristics will be given. This is relevant as transformational leadership theory augments that of transactional leadership according to Burns (1978).

#### Transactional Leadership Characteristics

In general, transactional leadership is categorised in the literature as a leader-follower relationship with an emphasis on reward and punishment, and bargains and exchanges to elicit desired behaviours (Dansereau et al., 2013; Gumusluoglu & Ilsev, 2009; Li et al., 2012). Knippenberg and Sitkin (2013) consider the transactional style a more traditional form of leadership compared with more contemporary outcome-based approaches such as transformational leadership. Transactional leadership contains three components, as defined by Bass (1997);

“**Contingent Reward** – Leaders engage in a constructive path-goal transaction of reward for performance. They clarify expectations, exchange promises and resources for support of the leaders, arrange mutually satisfactory agreements, negotiate for resources, exchange assistance for effort, and provide commendations for successful follower performance.

**Active Management by Exception** – Leaders monitor followers’ performance and take corrective action if deviations from standards occur. They enforce rules to avoid mistakes.

**Passive Management by Exception** – Leaders fail to intervene until problems become serious. They wait to take action until mistakes are brought to their attention.

**Laissez-Faire,** a nonleadership component, also emerges– Leaders avoid accepting their responsibilities, are absent when needed, fail to follow up on requests for assistance, and resist expressing their views on important issues”

Citing Lowe et al. (1996), Ewen et al. (2013) summarise that transactional leadership components have failed to demonstrate a consistently positive relationship with leader effectiveness, although contingent reward is positively associated in certain contexts, such as in public organisations. They go on to state that *“the ability to successfully achieve transformational leader outcomes is believed to require building on an effective transactional base”* (Lowe et al., 1996).

Bass and Avolio (1990) acknowledge that although often defined separately, transformational and transactional leadership can be displayed by the same leader. They also note in line with Lowe et al. (1996) that transactional leadership forms the basis for leader effectiveness which is subsequently augmented by transformational leadership. Lowe et al. (1996) suggest that in this view, transformational leadership is complimentary to transactional leadership, and would be ineffective without some form of transactional relationship between leaders and followers. In adding transformational characteristics, transactional contingent reward behaviours are improved and passive management-by-exception and laissez-faire leadership styles (which are perceived as ineffective leadership) are avoided (Bass & Avolio, 1990).

Transactional leadership therefore can be described as a ‘carrot or stick’ approach where leaders incentivise follower goal attainment through reward (carrot) or punishment (stick) (Bass, 1997). Reward and punishment is managed through the three components of transactional leadership; contingent reward (exchange of reward for performance), active management by exception (monitor and correct deviations) and passive management by exception (intervene after problem occurs). A fourth non-leadership component can also be seen – laissez faire (absent leaders).

#### Transformational Leadership Characteristics

Bass’ theory of transformational leadership consists of four components, often referred to collectively as “the four I’s of transformational leadership” (Antonakis & House, 2014; Avolio et al., 1991; Dansereau et al., 2013; Ewen et al., 2013; Li et al., 2015; Podsakoff et al., 1990; Qu et al., 2015; Van Knippenberg & Sitkin, 2013). These are; individualised consideration, intellectual stimulation, inspirational motivation and idealised influence (charisma) (Bass & Avolio, 1990).

Cho et al. (2011) suggested that IS success can be enhanced by transformational leadership through these components. For example, idealised influence - instilling pride, faith and respect, inspirational motivation - articulating an appealing vision and optimism, individualised consideration - coaching and mentoring, and intellectual stimulation - challenging users creative problem-solving skills. Li et al. (2012) describe a similar set of criteria, stating that a transformational leadership style involves idealised behaviour - charisma, inspirational motivation - stimulating optimism about the project, intellectual stimulation - promoting of new ways of thinking and solving problems, and individualised consideration - providing individualised care and consideration. These facets were found to effect followers in three ways: focusing attention on the importance and value of work being done, emphasising team goals over individual interests, and activating higher-order needs (Li et al., 2012).

What follows is a summary of the definitions for each of these four components and a discussion of how each dimension is portrayed in the literature.

##### Individual Consideration

Bass & Avolio (1990) argue that leaders elevate followers to take on greater responsibilities through individual consideration;

“The follower’s responsibility does not simply cover their job requirements, nor is it geared exclusively to maximising performance. On the contrary, followers are taking greater responsibility for their personal development, which includes such activities as one’s job challenges” (Bass & Avolio, 1990)

A large proportion of the literature reviewed, including van Knippenberg & Sitkin (2013), Qu et al. (2015), Avolio et al. (1991), Cho et al. (2011), Li et al. (2015) and Bass (1997) considered coaching and mentoring integral to the individual consideration dimension, suggesting that this allow leaders to help followers grow and personally develop. Qu et al. (2015) state that when leaders *“model empathy and support for individual concerns and openness to new suggestions and approaches…followers feel free to think in new ways, go beyond standard practices and proceed with creativity without fear of penalties”.*

Dansereau et al. (2013) and Podsakoff et al. (1990) include ‘showing concern’ and ‘respect’ for followers as integral facets of individual consideration, with Dansereau et al. (2013) going further to include caring for followers’ welfare and expressing appreciation and support for them. In line with this, Ewen et al. (2013) propose listening effectively and establishing a supportive climate as behaviours demonstrating individual consideration.

Qu et al. (2015) incorporate identification with the leader in their discussion of individual consideration, suggesting that it reduces role relationship uncertainty between leader and follower with positive effects. Avolio et al. (1991) and Cho et al. (2011) also recognise identification with the leader, however they attribute this to the idealised influence dimension which will be discussed later.

Across the literature the behaviours, traits and principles described by each author when discussing individual consideration share much common ground. Individual consideration hinges on the understanding that followers have individual needs which may differ from their colleagues and may change over time (Avolio et al., 1991; Bass, 1997; Bass & Avolio, 1990; Dansereau et al., 2013; Ewen et al., 2013; Qu et al., 2015; Van Knippenberg & Sitkin, 2013). Leaders demonstrating individual consideration understand this and seek to treat followers as individuals based on their differing needs, skills and aspirations (Bass, 1997; Qu et al., 2015). The literature suggests this may be approached by the leader personally coaching or mentoring followers (Van Knippenberg & Sitkin, 2013) creating a supportive and open climate (Ewen et al., 2013), showing empathy and support not only for work-related affairs but also for a followers family and personal life (Li et al., 2015), and showing a keenness to listen and share in followers concerns and needs.

##### Intellectual Stimulation

In much of the literature, discussion regarding the intellectual stimulation aspect of transformational leadership revolves around empowering individuals to problem-solve (Avolio et al., 1991; Bass & Avolio, 1990; Cho et al., 2011; Dansereau et al., 2013; Qu et al., 2015; Van Knippenberg & Sitkin, 2013). In conjunction with problem-solving, encouraging followers to challenge assumptions, beliefs, and traditions is often discussed. Bass (1997) argued that this stimulates and encourages expression of ideas and new ways of working, which enables more effective problem-solving.

Bass and Avolio (1990) emphasise the importance of challenging existing assumptions and fostering an environment where creativity, intuition and innovation are used to problem-solve. Avolio et al. (1991) expand on this stating that transformational leaders “*provide reasons why it is in the group’s best interests to work as a team to achieve its objectives”* and add that they encourage followers to question not only their own values and beliefs but those of the leader when confronting organisational problems.

Creativity is also frequently discussed as part of intellectual stimulation with Qu et al. (2015) stating that transformational leaders;

“Promote creativity by encouraging followers to question critical assumptions underlying the established framework of thoughts and routines and to look at old problems and situations in new ways”

Intellectual stimulation therefore is grounded in the notion that leaders *“inspire followers to develop their own innovative strategies”* (Guhr et al., 2019) by encouraging the expression of new ideas, challenging of assumptions, and empowering followers to be creative in problem-solving based on reasoning and evidence. This is seen to positively impact employee motivation as they feel their opinions matter and their voices are heard, this in turn leads to more faith in the leader (Avolio et al., 1991).

##### Inspirational Motivation

Inspirational motivation is defined by Li et al. (2015) as *“the degree to which a leader articulates an appealing vision to followers, communicates optimism about future goals, and provides meaning for the task at hand”*. Across the literature this definition was common, with Van Knippenberg and Sitkin (2013) suggesting that uniting followers around a clear vision and mission created a sense of collective purpose.

Cho et al. (2011) added that leaders who demonstrate inspirational motivation express high levels of expectation and optimism, which aligned with Bass (1997);

“[Transformational leaders] articulate an appealing vision of the future, challenge followers with high standards, talk optimistically with enthusiasm, and provide encouragement and meaning for what needs to be done”

There was also an element of example setting given by Avolio et al. (1991) who suggested leaders inspire motivation through “pep-talks” and seek to relive the burden of their followers by striving for creative work methods to lessen the load.

Inspirational motivation is therefore demonstrated through expressing a clear and articulate vision, uniting followers around a common goal and demonstrating a belief and optimism that followers can achieve their collective mission.

##### Idealised Influence (Charisma)

Idealised influence is defined as communicating an inspiring vision and high expectations of performance (Van Knippenberg & Sitkin, 2013). In the literature reviewed, charisma was frequently given as a descriptor of idealised influence (Avolio et al., 1991; Li et al., 2015; Van Knippenberg & Sitkin, 2013). Van Knippenberg and Sitkin (2013) noted that idealised influence and inspirational motivation are often combined into a single “charisma” factor due to the overlap between vision, mission, and collective sense of purpose. Avolio et al. (1991) had similar observations and suggested that;

“Idealised influence is, to a large degree, a culmination of the other three I's coupled with a strong emotional attachment to and identification with the leader. Such leaders are emulated by followers and often labelled charismatic”

Bass and Avolio (1990) suggested that through idealised influence followers were seen to identify more closely with their leaders and consequently worked harder to achieve optimal performance. Cho et al. (2011) suggested that identification with the leader was achieved by demonstrating admirable behaviours which tapped into followers emotions. Podsakoff et al. (1990) added that a key aspect of idealised influence was the leader appearing to be above self-interest, thereby demonstrating the values and behaviours essential in followers to fulfil the organisational mission.

There was also a theme of instilling a sense of the ‘greater good’ for the group rather than individual reward (Avolio et al., 1991; Bass, 1997; Ewen et al., 2013; Li et al., 2015; Podsakoff et al., 1990). Li et al. (2015) go as far as defining idealised influence as “*the degree to which a leader shows charisma and emphasizes the importance of purpose, commitment, and the ethical consequences of decisions”.*

Leaders who demonstrate idealised influence or charisma motivate their followers to emulate and follow in their footsteps. This is done by identifying more closely with followers, demonstrating and embodying the values and behaviours expected to achieve the vision, and shaping a mission, through which positive outcomes are perceived for all rather than the few.

This chapter has introduced the subject of leadership in general and has reviewed the existing literature on the topic of leadership in IS. It has also defined terms and examined transformational-transactional leadership theory, which was found to be dominant in current research. Definitions of leadership in the literature were found to be wide ranging and, in some cases, vague. One undertaking for this review therefore was to define terms as they will be used in this study. Table 11 provides a summary of these terms.

In the following section, parallels between agile principle and the characteristics of transformational leadership theory will be discussed, demonstrating how this aspect of the literature led to the selection of this theory as a lens for this study.

Table 11: Terms defined through review of leadership literature

|  |  |
| --- | --- |
| **Key Term** | ***Definition*** |
| **Leadership** | *Leadership is the ability of an individual to influence others through activities of direction, support, and inspiration. Leaders provide direction through uniting people around a common vision. They support through empowering and trusting those they lead to do what is needed to achieve the shared vision. They inspire through providing followers with purpose, leading by example, and demonstrating shared values and beliefs* |
| **Transformational Leadership** | *Empowering and intellectually stimulating individuals to seek a higher order of aspirations. Aspirations which focus on team and organisational goals, not just individual desires* |
| **Individual consideration** | *Seek to treat followers as individuals based on their differing needs, skills and aspirations. Show a keenness to listen and share in followers’ concerns* |
| **Intellectual stimulation** | *Inspire followers to develop their own innovative strategies. Encourage the expression of new ideas, challenging of assumptions, and empowering followers to be creative in problem-solving* |
| **Inspirational motivation** | *Express a clear and articulate vision, uniting followers around a common goal and demonstrating a belief and optimism that followers can achieve their collective mission* |
| **Idealised Influence (Charisma)** | *Motivate followers to emulate and follow in leaders’ footsteps. Identifying with followers, demonstrating and embodying the values and behaviours expected to achieve the vision. Shape a mission, through which positive outcomes are perceived for all rather than the few* |
| **Transactional leadership** | *A leader-follower dynamic with an emphasis on reward and punishment to elicit desired behaviours* |
| **Contingent Reward** | *Leaders engage in a constructive path-goal transaction of reward for performance. They clarify expectations, exchange promises and resources for support of the leaders, arrange mutually satisfactory agreements, negotiate for resources, exchange assistance for effort, and provide commendations for successful follower performance* |
| **Active Management by Exception** | Leaders monitor followers’ performance and take corrective action if deviations from standards occur. They enforce rules to avoid mistakes |
| **Passive Management by Exception** | Leaders fail to intervene until problems become serious. They wait to take action until mistakes are brought to their attention |
| **Laissez-Faire** | A nonleadership component – Leaders avoid accepting their responsibilities, are absent when needed, fail to follow up on requests for assistance, and resist expressing their views on important issues |

#### Agile and Transformational Leadership

The Agile Manifesto evolved in response to increased competition, unpredictable markets, rapidly advancing information technologies, an unprecedented rate of change, and the need for firms to evolve in turbulent environments (Chakravarty et al., 2013; Lee & Xia, 2010; Li et al., 2015; Nazir & Pinsonneault, 2012; Vidgen & Wang, 2009). It is predicated on the notion that in a world of fast-paced change and volatile markets, the future is unpredictable. It suggests therefore to focus not on trying to eliminate change but to embrace or even use it to your advantage. To do this the manifesto proposes a set of values and principles on which a number of software development methodologies are based. In brief, agile values people over process, responding to change over following a plan, collaboration over contract negotiation and delivering value over detailed documentation. Its principles focus on eliminating waste, learning through continuous improvement, delivering as fast as possible and empowering teams.

As organisations have evolved in response to these changes in their environments, so too have the concepts and perceptions of leadership;

“Modern organizations have changed significantly in size and complexity over time and so also have the skills and styles of leadership required for the modern-day organisation.” (Ola, 2017)

It is clear therefore that in conjunction with changing their approach to delivery, organisations must also consider how best to lead in complex and ever-changing environments.

Transformational leadership shares a number of commonalities with agile values and principles. For example, in agile there is an emphasis on providing the environment and support the team needs and in trusting them to get the job done. This has parallels with transformational leadership through the **individual consideration** component which encourages leaders to mentor or coach individuals and create a supportive environment. It also mirrors aspects of **inspirational motivation** in terms of showing optimism that followers can achieve their goals.

In Scrum (an agile methodology), setting an inspiring and clear goal or mission for the team to rally around is paramount. This is also a central tenet of transformational leadership where expressing a clear and articulate vision and uniting followers around a common goal are aspects of the **inspirational motivation** and **idealised influence** components. The team retrospective in Scrum, a forum where issues and concerns are addressed, also shares commonalities with **individual consideration** where leaders show a keenness to listen and share in followers concerns and needs.

Finally, agile working flattens hierarchy and allows all members of the team to operate on a common ground, where all voices can be heard and ideas freely shared. Similar values are seen in the **intellectual stimulation** component of transformational leadership which encourages the expression of new ideas, challenging of assumptions - even those of the leader, and empowering followers to be creative in problem-solving.

As demonstrated, there is significant overlap between agile values and principles and the components of transformational leadership. For this reason, transformational-transactional leadership theory (Bass, 1985) was chosen as a lens through which to explore the phenomena under study – the adoption of agile working practices. In the following section a summary of the literature reviewed in this chapter will be presented along with gaps which help to justify the focus of this study.

## Summary

Our understanding of leadership has evolved over the decades it has been studied, moving away from the concept of ‘a great man’ who was destined to become a leader and towards a trait-based view where the characteristics and styles of leadership in general have been explored. This review has indicated that although the study of leadership in general has had a resurgence in recent years, there remains a lack of focus on leadership in the IS domain, where leaders have IS specific concerns.

Of those studies which did take a leadership perspective within IS, a number of common themes were found. Namely; emergent leadership within dispersed teams, the role of leaders in the successful rollout and adoption of information systems, and the influence of leaders on the creativity and motivation of teams. There was also a subset of papers dedicated to literature reviews. Notably however, studies have been scarce on the influence of leadership in the context of agile adoption.

Much of the literature reviewed adopted a quantitative (survey) approach to studying leadership, limiting the richness of the data. Although this approach has moved our knowledge forward in broad terms, there is a lack of understanding regarding the impact of leadership on practice, such as the day-to-day adoption of agile. Dubinsky & Hazzan (2010) began to address this through their research. They worked as consultants to their cases and, as such, could carry out longitudinal case studies. However, they stated that their engagement was external, involving 15-20 hours per month to provide training and facilitation. Consequently, they could not provide an insider perspective. They presented a model, ad-hoc leadership, through which to understand the different relationships and interactions which occurred between the change leader in an agile transition and other central groups – the management, the team, and the customer. However, they did not explicitly explore leadership styles as they pertained to agile transition and adoption, despite the name of their model. This appeared to refer only to the fact that the change leader changed their approach based on who they were interacting with, and hence adopted an ad-hoc approach to leading the agile transition.

There have been calls from scholars for more studies focused on how different styles of leadership may be suited to different sectors (Ghawe & Brohman, 2016), for leadership to be studied outside of the traditional organisational behaviour domain (Behrendt et al., 2017; Van Knippenberg & Sitkin, 2013), for researchers to understand more about how leaders influence process (Dinh et al., 2014), and to take a longitudinal perspective of the phenomena (Dinh et al., 2014). This study addresses each of these gaps, as will be described below.

First, this study focused on leadership within agile software development and, as such, answers calls to explore how different styles of leadership may be suited to particular sectors. Second, focusing on leadership within the IS domain and specifically within software development, this research addresses calls to study leadership outside of the traditional organisational behaviour domain. Third, this study concentrated on the impact of leaders on the process of agile software development and therefore addresses the call to understand more about how leaders influence process. Finally, through the adoption of a qualitative and longitudinal approach, this research provides richness on the phenomena under study. This is given from an embedded perspective thanks to the immersed position of the researcher at SoftwareCorps. Maximising on this, a thick description of the phenomena is provided, focusing on the journeys, behaviours and experiences of the participants. By being immersed within the case, a deep understanding of the context in which the phenomena was studied is offered, which current research is yet to provide. This addresses criticism from scholars regarding a lack of qualitative longitudinal research in the IS literature.

Perhaps most significantly for this study are the gaps highlighted by Strode (2016) and Mishra et al. (2017) regarding a lack of knowledge on the variation apparent in how agile is adopted, and in the lack of research on leadership in agility. This study takes a leadership perspective on the adoption of agile working practices in software development teams, helping to understand how leadership influences variation in adoption. In addition to this, by applying leadership theory to the agile domain, this study addresses critiques by scholars that the agile domain is typically a-theoretical. Findings will be presented in Chapter 7 demonstrating the how the leadership styles of two individuals, Nelson and Julius, differed. The influence of each leader on the adoption of agile working practices will be explored through the behaviours of those leaders, and subsequently the behaviours of their followers (software development teams). This perspective, whilst predominantly focusing on leadership style, also builds on the concepts of leadership characteristics, behaviours and philosophies in the existing literature to tell a rich and descriptive story of the phenomena under study.

Before presenting the findings, the following chapter details the research methodology selected to carry out this study.

# Research Methodology

The purpose of this chapter is to introduce and justify the research methodology selected for this study of agile adoption in software development teams. The research methodology describes the ontological and epistemological commitments made by the researcher. The research method describes the techniques and procedures used to collect and analyse data (Prasad, 1997). Both will be discussed in this chapter. First, the research objectives of this study are reviewed, demonstrating how they contributed to the selection of an appropriate research methodology and method. Next, a brief overview of philosophical assumptions underpinning the research will be presented along with justification for those chosen. Finally, details are provided regarding the selected research methods, including research design, case selection, data collection methods, and data analysis.

## Research Aim

The overarching aim of this study is to understand how agile working practices and principles are adopted in software development teams. This aim sought to address a lack of knowledge surrounding the variability with which agile methods are adopted. In order to fulfil this aim, the following research questions were posed;

**RQ1:** How do the practices of agile software development enable the manifestation of agile principles in software development teams?

After an initial phase of participant observation and interviews, early analysis indicated leadership was a dominant theme, and therefore a second research question was added;

**RQ2:** How does leadership style affect the adoption of agile working practices in software development teams?

Software development is a complex socio-technical activity involving team interactions, behaviours and organisation, as well as technical methods and techniques (Sawyer, 2004). The Agile Manifesto is predicated on a set of principles and values rather than a prescriptive set of practices. There are, however, methods which sit under the agile methodology which provide these practices, such as Scrum. It was necessary, therefore, to select a research methodology that would allow for an in-depth exploration of the social and cultural aspects of agile adoption. Careful consideration was also given to the philosophical perspective from which this study would be carried out, given the objectives of the research. The philosophical underpinnings of the research and the methods selected are discussed in the following sections.

## Research Philosophy

Philosophical assumptions on which research methods are based are commonly divided into three categories; interpretivist, positivist and critical (Myers, 1997). The interpretivist view is generally most suited to the exploration of a research topic or theory, the positivist view to testing hypotheses, and the critical view to critiquing the status quo or helping to emancipate people in some way (Myers, 1997). Traditionally, qualitative research methods adopt an interpretivist perspective while quantitative methods adopt a positivist perspective, although this is not always the case. For example, there are positivist, interpretive, and critical examples of both action research and case study research (Klein & Myers, 1999).

Although in the past information systems research was dominated by positivist quantitative studies, it is now commonplace to see the application of more diverse research methodologies and philosophies, including interpretivism (Walsham, 2006). This diversity is, in part, thanks to a shift towards exploring managerial and organisational aspects of information systems, on which an interpretivist perspective is informative (Myers, 1997). When determining the philosophical assumptions for this study, significant consideration was given to the research objectives and aims, resulting in the selection of an interpretivist perspective. This perspective is discussed in more detail in the following section.

### Interpretivist Perspective

It is essential to first distinguish between interpretive research and qualitative research, as these terms are sometimes erroneously used synonymously (Conboy et al., 2012). In this chapter, interpretivism refers to the researchers underlying philosophy, whereas qualitative refers to the research methodology (Klein & Myers, 1999). Interpretive researchers attempt to understand phenomena through the meanings people assign to them and the context within which they exist, focusing on the complexity of human sensemaking (Kaplan & Maxwell, 1994; Myers, 1997). Within IS, research can be considered interpretive when knowledge is perceived as being gained through social constructs, including language, consciousness, shared meaning, documents, tools, and artefacts (Klein & Myers, 1999). Interpretive approaches allow for a greater understanding of human and organisational phenomena when compared with their positivist counterparts, emphasising lived experience as the basis of human action and activities (Sandberg, 2005).

As highlighted in Section 4.1, the objective of this study was to gain a greater understanding of the adoption of agile practices and principles in software development teams. Given the socio-technical nature of software development and the focus of the Agile Manifesto on values and principles, an interpretive perspective was appropriate. The researchers’ perspective allowed for the experiences and narratives of the participants to be elicited. It also allowed and for an in-depth exploration of the journey of agile adoption within SoftwareCorps, observing the phenomena in a real-life context. In the following section, the choice of a qualitative research approach for this study will be presented.

## Research Approach

Research methods are commonly classified as either quantitative or qualitative, with some researchers opting for a combined mixed-methods approach (Myers, 1997). This section will briefly describe quantitative and qualitative methods and justify the choice of a qualitative approach for this study.

### Quantitative

In basic terms, a quantitative approach seeks to statistically or experimentally test a hypothesis, removing data from its context in order to provide objectivity and generalisability (Kaplan & Duchon, 1988). As such, more focus is given to testing theories and hypothesis, rather than exploration. Examples of quantitative tools are; survey, laboratory experiments, and mathematical modelling. A quantitative approach is most suitable where large data samples are being handled, and there is a desire to generalise trends and patterns to a broader population (Creswell, 2003, p. 18; Myers, 2009, p. 9). However, this can lead to a lack of depth and understanding regarding what is actually occurring within the context of the data being studied (Kaplan & Duchon, 1988). The information systems field is traditionally dominated by positivist quantitative research (Galliers & Huang, 2012). However, interest in qualitative research and an interpretivist perspective has grown (Goldkuhl, 2012). It has been acknowledged that reducing complex IS social phenomena to quantitative figures can be limiting and that qualitative methods allow more open and nuanced exploration of these phenomena (Goldkuhl, 2012; Kaplan & Maxwell, 1994; Myers, 1997).

### Qualitative

Qualitative methods rely on context to provide a rich and deep understanding of complex phenomena in a natural setting (Myers, 1997). This context provides an opportunity for researchers to explore why and how these phenomena occur and to understand social and human problems. In line with this, qualitative tools include interviews and observation as well as analysis of text and documents. A qualitative approach is most suited to smaller data sets or single cases, especially where a longitudinal perspective is desirable. However, this can limit the generalisability of findings.

Qualitative methods have become a robust and valuable tool in furthering our understanding of how and why phenomena happen, and in understanding human interaction and meaning. Although qualitative methods are now more widely used in IS research (Conboy et al., 2012), it should be noted that there is a relative lack of qualitative research published in top journals (Galliers & Huang, 2012).

### Mixed-Methods

The development of pluralist mixed-methods approaches sought in part to address the deficiencies identified when adhering to a single quantitative or qualitative approach. Mixed-method researchers select tools as appropriate from across qualitative and quantitative methods in order to provide a thorough investigation of phenomena under study (Teddlie & Tashakkori, 2012). In comparison to quantitative and qualitative methods, this is a relatively new field and as such debate remains regarding the appropriateness of mixing methods and philosophies within a single study (Teddlie & Tashakkori, 2012).

### Choice of Research Approach

Driven by an interpretive philosophical underpinning and guided by the research objective of this study to understand how the adoption of agile manifests in a real-life context, it was determined that a qualitative research approach would be most suitable. As discussed in Chapter 2, gaps were identified surrounding how agile methods are adopted over time and through periods of organisational change. A qualitative approach supported this objective in two ways. First, it allowed a deep and rich description of the journey taken by teams in an organisation transitioning to agile working. This addressed criticisms by scholars that there was a lack of detail on the subject in the extant literature. Second, it allowed the researcher to take an immersive role within the organisation, enabling exploration of the research objectives in a real-life context. This allowed a greater understanding of the social and human aspects of the phenomena. In the following section, the research strategy and choice of a case study research method are discussed.

## Research Strategy

There are many methods to consider when carrying out qualitative research, each of which influences the ultimate research design and data collection strategy (Myers, 1997). Selection of an appropriate research method must be driven predominantly by the research question being posed, as well as constraints such as access to data. Two commonly used methods; case study and action research, will be described in this section before justification for the choice of method used in this study is given.

### Case Study

The term ‘case study’ can apply both to a research method (as will be described here) and unit of analysis, such as an organisation (Myers, 1997). As a research method, the case study is one of the most widely used in agile software development studies (Chuang et al., 2014) and is frequently seen in information systems research (Myers, 1997). According to Yin (2013), the case study is an in-depth inquiry into a specific complex phenomenon in its real-world context, where an examination of the interaction between the case and its context is paramount. Yin (2014, p. 4) explains that the case study method is most suitable where researchers seek to explain the how or why of a phenomenon, or where research questions require an in-depth description of the phenomenon. Yin (2014, p. 4) states that the need for case studies arose from a desire to understand complex social phenomena, exploring them in-depth while retaining a real-world perspective.

### Action Research

Action research places the researcher in an active 'helping' role within the organisation being studied. It involves the collaborative analysis of the social situation by both the researcher and subjects of the research, generating hypotheses, and collaboratively introducing change to study its effects (Baskerville & Wood-Harper, 1996). Typically, the focus of action research is on the change process itself and lines are blurred between the researcher and those being researched (Checkland & Holwell, 1998). In this way, action research contributes not only to scientific knowledge but also to the practical concerns of those being studied (Myers, 1997). Action research is most suited to the study of complex human processes, where the aim is to enhance the development of models and theories (Baskerville & Wood-Harper, 1996).

### Choice of Research Method

In this study, the unit of analysis is a single case (SoftwareCorps) with the application of a case study research method. The case study approach allowed the researcher to gain an in-depth understanding of the phenomena from the perspective of those experiencing it. This approach complemented both the interpretive philosophical perspective and the objectives of the study. In addition to this, the nature of the research questions, seeking to understand the ‘how’ of a particular topic, suited the selection of case study as a research method.

Action research was also considered, however, due to the exploratory nature of the research objectives it was determined that a case study would be more suitable.

## Research Design

This section presents the overall research design, research procedures and details of how this case study was carried out. A two-phase approach was taken; During phase one, the adoption of agile practices and principles, and factors influencing this adoption were explored. Early analysis of phase one data informed an additional research question which led to phase two of the study. This focused on how leadership style influenced the adoption of agile practices and principles.

According to Yin (2014, p. 29) there are five important components of case study research design;

1. a case study’s questions;
2. it’s propositions, if any;
3. its unit(s) of analysis;
4. the logic linking the data to the propositions; and
5. the criteria for interpreting the findings.

##### Case study questions

Questions typically suited to the case study method are those which ask ‘how’ and ‘why’ questions (Yin, 2014, p. 29). Three stages of question development were followed for this study, as recommended by Yin (2014, p. 29). First, the literature was searched to narrow the interest to a key topic. Second, key papers on the topic of interest were examined more closely and their suggestions for future research explored to help define early research questions for this study. Third, the researcher returned to the literature with these potential questions to reinforce and sharpen them.

##### Case study proposition

Where necessary, the proposition of a case study directs attention to something of interest to the scope of the study, pointing towards *what* should be studied (Yin, 2014, p. 30). Where a study seeks to ‘explore’ a topic, there may not be an explicit proposition, however a clear purpose must be defined (Yin, 2014, p. 30). For this study, an initial exploratory phase was carried out with the purpose of understanding how agile practices help to manifest agile principles in software development teams, and factors influencing the adoption of these practices. During the second phase of the study, focus was narrowed to a specific proposition – how leadership style influences the adoption of agile practices.

##### Unit of analysis

There are two considerations with regards to the unit of analysis in case study research; defining the case and bounding the case (Yin, 2014, p. 31). These considerations are derived predominantly from the research questions and case study propositions which should direct the choice of the unit (Yin, 2014, p. 32).

A single case was selected for this study; SoftwareCorps. In bounding the case, considerations of the topic and context for the study were considered in order to distinguish those persons who should be included and those who were outside of the case (but provide the context of the case) (Yin, 2014, p. 33). The boundaries set for this study included focusing on those individuals who contributed to the development of software within the organisation, predominantly those within the software development department with select exceptions such as executive managers and product owners. Selection of participants for this study is discussed in detail in Section 4.5.4.

##### Linking data to proposition

It is helpful to consider the analytical techniques which may suit the research proposition and help to inform the research questions during the case study design phase (Yin, 2014, p. 36). Data may be linked to propositions through a number of techniques including pattern matching, explanation building, time series analysis, logic models and cross-case synthesis (Yin, 2014, p. 36). Yin (2014, p. 36) acknowledges that where researchers have limited experience in empirical studies, identifying these techniques upfront may be difficult and potentially limiting. For this study, interview data was identified as the primary source of data and a variety of qualitative data analysis techniques (based on recommendations from Miles, Huberman and Saldaña (1987)) were considered. This is described in more detail in Section 4.6.

##### Criteria for interpretation of findings

In quantitative studies, significance of findings are generally identified statistically (Yin, 2014, p. 36). In case study research, in order to determine whether findings are significant, researchers do not rely on statistics. Instead they must consider rival explanations of their findings in order to ensure data about them is included as part of data collection (Yin, 2014, p. 36). Addressing these rivals becomes a criterion for interpreting the findings of the study (Yin, 2014, p. 36). This component forms part of the data analysis design along with the previous component, linking data to propositions.

### Credibility

Although it is arguably impossible to achieve complete validity and reliability in any research, regardless of its type, it remains an essential task for researchers to conscientiously consider these issues in their research design to ensure credibility of findings (LeCompte & Goetz, 1982). Yin (2014, p. 45) identifies several tactics specific to case study research which deal with the four tests of research design;

Table 12: Case Study Tactics for Four Design Tests (adapted from Yin, 2014, p. 45)

|  |  |
| --- | --- |
| Test | Case Study Tactics |
| **Construct Validity**  *Identify correct operational measures for concepts being studied* | * Use multiple sources of evidence * Establish chain of evidence * Have key informants review draft case study report |
| **Internal Validity**  *For explanatory or causal studies only, establish a causal relationship and distinguish from spurious relationships* | * Do pattern matching * Do explanation building * Address rival explanations * Use logic models |
| **External Validity**  *Define domain to which findings can be generalised* | * Use theory in single-case studies * Use replication logic in multiple-case studies |
| **Reliability**  *Demonstrate the operations of the study are repeatable, with the same results* | * Use case study protocol * Develop case study database |

#### Validity

The validity of case study research concerns the accuracy of the findings. A common criticism of case studies is that there may be a failure to develop a sufficiently operational set of measures and that, consequently, subjective judgements are used to collect the data (Yin, 2014, p. 46). To mitigate against this, Yin (2014, p. 45) suggests a number of tactics to address construct, internal and external validity, as can be seen in Table 12.

Two important steps must be taken to meet the test of **construct validity**; First, concepts must be clearly defined and related to the objectives of the study. Second, operational measures that match these concepts must be identified, preferably citing published studies making the same matches (Yin, 2014, p. 47). Concepts for this study, including agile, leadership, methodology, principles and practices have been clearly defined through the literature reviews. Subsequently, measures used to identify agile principles (mapping practices to principles) and identify leadership styles (transformational and transactions) were also defined based on those in published studies. In addition to these two steps, construct validity was achieved through the use of multiple sources of evidence (interview, participant observation and documentation), creating a chain of evidence whereby conclusions can be traced back to supporting evidence supported by a detailed description of the methods used to collect data, and by reviewing early data displays and theories with participants. These are discussed in more detail in Sections 4.5.5 and 4.5.6.

The test of **internal validity** is less relevant for this study, which sits within the exploratory/descriptive category rather than explanatory/causal. However, the broader problem of inferences can be extended to this test (Yin, 2014, p. 47). For example, when an event cannot be directly observed, the researcher may infer that a particular event resulted from some earlier occurrence. When designing the case study these inferences must be recognised, and internal validity strengthened by incorporating tactics such as addressing rival explanations as part of the study design.

**External validity** concerns the generalisation of results from a case study (Yin, 2014, p. 48). This study deals with analytic generalisation (based on theory) rather than statistical generalisation which assumes a ‘sample’ is representative of the population (Yin, 2014, p. 40). Analytical generalisations are made based on corroborating, modifying, rejecting or advancing existing theory, or by creating new theories, and are at a conceptual level higher than the case itself (Yin, 2014, p. 41). The existing theory of transactional and transformational leadership was advanced through this study and the application of that theory to the adoption of agile practices and principles. As this theory has been explored heavily in the management literature, this supports the generalisation of findings from this study outside of the case itself. Suggestions for future research in Section 9.3 illustrate how this generalisation might be carried out.

#### Reliability

The reliability of case study research concerns the replicability of findings (Yin, 2014, p. 48). It should be possible for a later researcher to follow the same procedures described in a study at the *same case* and arrive at the same findings and conclusions (Yin, 2014, p. 48). Methods of data collection and analysis must therefore be made explicit. Ideally, methods should be described in such a way that other researchers could follow the report as a manual for replication. This level of detail is often not possible given the constraints of format for publication (such as journals), however, every effort must be made to provide *sufficient* detail on the methods of data collection and analytic strategy used (LeCompte & Goetz, 1982). Both tactics suggested by Yin (2014, p. 45) and presented in Table 12 to increase reliability were used in this study; case study protocol (see Section 4.5.6) and case study database (NVivo).

What follows in this chapter is a detailed report of how this research was carried out which seeks to tackle concerns of credibility as described in this section. Additional concerns for case study researchers are maintaining ethical and moral standards, as will be discussed next.

### Ethics

Ethics in qualitative research is defined by Myers (2009, p. 45) as the application of moral principles in carrying out all aspects of research. He adds that this includes respecting and protecting those who have actively consented to be studied. Maintaining ethical standards is paramount in case study research. The researcher is in a position of trust and how the researcher represents and interprets the stories of those they study must, therefore, be carefully considered.

As advised by Myers (2009, p. 47), the following ethical issues were considered for this study;

* Privacy - Ensure confidentiality is preserved
* Representation – Honestly report and analyse data
* Responsibility – Stand behind findings, even if they are unfavourable

Privacy was upheld in this study through the anonymisation of all names and any identifying data. It was made clear to the participants during interviews and observations that any data collected would be anonymised and used for the sole purposes of the research project, not wider organisational ends.

To ensure an honest representation of the participants’ stories and experiences in this study, they were provided with full transcripts and invited to review them. The participants were encouraged to provide clarification or corrections if they felt this was necessary. The researcher also reviewed early visualisations of the data, such as timelines, with the participants to ensure these were representative.

Written informed consent was requested from every participant who was interviewed, making the use and purpose of data collected explicit before interviews were carried out. In addition to this, it was widely communicated that observations were being carried out for PhD research. Permission was obtained from the chief technology officer (CTO) for participant observation to be carried out with the understanding that data would be anonymised.

### Identify a Research Problem

LeCompte and Goetz (1982) state that *“formulation of an initial research problem involves both the delineation of a content area and the choice of appropriate design and methods for investigation”.* Both of these considerations drove the identification of a research problem for this study. First, it was evident there was a gap in the existing body of knowledge regarding how transitioning to agile working should be approached and factors influencing success. Second, the researcher had access to an organisation undergoing such a transition. As discussed in Chapter 2, an initial literature review was carried out which identified a lack of research regarding how organisations transition to agile ways of working. This gap led to the definition of the research problem for this study – to understand more about how agile principles and practices are adopted day-to-day.

### Selection of the Case

The researcher had an existing relationship with contacts at SoftwareCorps, which later became the case for this research. Having restructured their management team and introduced a new CTO (see Chapter 5 for more details), SoftwareCorps were transitioning to an agile way of working. The circumstances at SoftwareCorps provided an ideal opportunity for an in-depth qualitative study to be carried out.

Having identified this opportunity, the researcher first sought permission from the CTO, who was leading the agile transition. With the support of the CTO, the wider management team were approached. Access was granted on the basis that participation in any data collection would be voluntary and would not interfere with the day-to-day work of participants.

#### Selection of the Participants

The choice of participants for this study was driven by the research questions, which seek to better understand how software development teams adopt agile practices. Guided by this, participants from the software development and product teams, including executive managers (EX), lead developers (TL), developers (D), testers (QA), scrum masters (SM), and product owners (PO) were invited to participate to provide a cross-section of views and experiences. It was deemed necessary to interview individuals from a variety of levels within the organisation, and to select individuals with different levels of seniority and tenure, to provide a diverse set of perspectives (Myers, 2013, p. 126). When designing the inclusion criteria of participants for interview, consideration was given to the fact that qualitative research does not necessarily seek to provide a 'representative' set of participants (Mason, 2010). Instead, it relies on telling the stories of participants who are willing and open to sharing their own opinions and experiences with the researcher (Mason, 2010), providing an in-depth and rich view of the subject under study.

Therefore, the following conditions were considered in the selection of interview participants;

* Providing a spread of participants in roles from across the software development population
* Including participants known to be actively for, actively against, and indifferent to agile practices in order to capture a spread of views
* Availability of the participant (for example, some teams may be unable to participate due to project deadlines)
* Mitigating the possibility that requests for interviews would be declined by extending invitations for participation to as many individuals as possible

Across the roles identified, taking into consideration the criteria above, 68 individuals were identified. It was not feasible or realistic to set up interviews with all 68 individuals identified from the outset. Instead, an iterative approach was taken, and invites were distributed in batches prioritising individuals who met with the inclusion criteria stated above. Table 13 shows the total number of individuals available in each category in brackets, along with the total number of participants ultimately interviewed in each group for each phase.

Table 13: Cross Section of Participants Interviewed

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Participants[[3]](#footnote-3):  Phase & RQ | EX (3) | SM (4) | TL (8) | PO (6) | D (34) | QA (13) | Total |
| Phase 1: How is agile adopted? | 3 | 4 | 7 | 5 | 6 | 2 | 27 |
| Phase 2: How does leadership style affect adoption? | 3 | 2 | 6 | 2 | 5 | 2 | 20 |
| Total | 3 | 4 | 7 | 5 | 7 | 3 | 29[[4]](#footnote-4) |

In total, 29 participants were interviews for this study. In phase one, 27 interviews were carried out. In phase two a further 20 interviews were carried out. Nine of the participants from phase one were not interviewed in phase two due to extenuating circumstance; five left the company before phase two began, the others declined to participate further due to project commitments. Two participants (D, QA) were unavailable during phase one due to project commitments and so participated in phase two only.

Of all the participants who were invited to take part in the study, one product owner and one lead developer were non-responders. Except for one lead developer, who withdrew from phase two due to project commitments, all other participants accepted invitations to participate in an interview.

### Data Collection Methods

In a case study, a period of on-site fieldwork is carried out where data are collected primarily through interview (Myers, 1999). It is the intervention of the researcher in this collection, selecting some things and excluding others, that transforms *potential data* into data that holds relevance and forms the basis of research (Wolcott, 1994, p. 3). Fieldwork requires respecting the daily lives of the participants, maintaining sensitivity to them, reciprocity (such as providing them with advice), and maintaining ethics in all aspects of the research (for example being explicit about the research undertaking) (Creswell, 2007, p. 71). In this section, an overview of the types of data collection methods used in this study will be given.

#### Interviews

Interviews are the most common form of data collection in qualitative research, allowing the researcher to gather rich data from a variety of people (Myers, 2013, p. 119). There are several interview types, typically categorised into structured, semi-structured and unstructured (Myers, 2013, p. 121). Structured interviews use pre-defined questions which are strictly adhered to, providing consistency across interviews. Unstructured interviews have few if any pre-defined questions, the participant is free to say whatever they want, and the interviewer is expected to improvise. Semi-structured interviews are a combination of structured and unstructured interviews. They have some pre-defined questions (usually open-ended), and new questions emerge during the conversation, which is encouraged (Myers, 2013, p. 122). Semi-structured interviews take the best parts of structured and semi-structured interviews while minimising risk. For example, important data could be missed through adherence to pre-defined questions, or a participant may be un-talkative in an unstructured interview (Myers, 2013, p. 122). A semi-structured interview provides the opportunity for the researcher to act on insights during the interview, based on the participant’s answers. For these reasons, semi-structured interviews were selected for this study.

#### Participant Observation

Participant observation involves the researcher observing people in their natural setting. The researcher is immersed in the ongoing activities of the group and is personally involved in order to gain an understanding that will be shared with others (Myers, 2013, p. 136). There is an important distinction between observation, where the researcher observes from the outside, and participant observation, where the researcher participates in the activities they are observing. This distinction allows the researcher to gain a deeper understanding and an insider perspective of the society and culture they are observing (Myers, 2013, p. 137).

#### Documents

Documents were used in this study as a supplementary or supportive form of data rather than as a central source of insight (Myers, 2013, p. 159). For example, documents were used to supplement the history provided during interviews of the software development processes used historically. Most documents collected were already in electronic form, where this was not the case electronic copies were taken using a scanner or by photographing the document.

### Data Collection Process

This section will detail the processes and procedures of data collection carried out during this study, based on advice and guidance provided by experts in the field, as discussed in the previous sections.

#### Pilot Interviews

Initially, a small number of pilot interviews were carried out to test the semi-structured questions, interview procedures, and interview protocol. From these pilot interviews the following protocols were defined;

* All individuals invited to participate in research interviews were advised that involvement was voluntary and that any data collected would be anonymised (see Appendix 11.1).
* Participants were provided with a consent form at the start of the interview detailing how the data from the interview would be used and collecting basic demographic information (see Appendix 11.2).
* Interviews were carried out on-site in meeting rooms which were familiar to the participants.
* Interviews were booked for 1 hour, with rooms booked for an additional 30 minutes to minimise any disruption if the interview over-ran (at the discretion of the participant).
* Participants were not provided interview questions in advance of the interview as the participants in pilot interviews did not read them when given a copy in advance.
* With the participant’s permission, interviews were recorded using an iPhone and lapel microphone for transcription at a later date (see Section 4.6.1 for details on transcription process).
* With permission from the participant, notes were taken throughout the interview to mitigate the risk of problems when recording the interview.

#### Semi-Structured Interviews

Two phases of interview were carried out for this study. In both phases, interviews were designed to capture data on actual events in order to present the story of the case and to capture the participants’ perspectives regarding their experience of those events (Yin, 2014, p. 102). Interview questions were positioned in a way to explore the research objectives and the participants’ experiences in the adoption of agile working. An informal and conversational approach to the interviews was developed during the pilot interviews (discussed next).

##### Designing the interview instruments

For the phase one interviews, an instrument was developed to guide the conversation on the topics of agile adoption, the use of agile practices, and views of agile principles. Four versions of this instrument were created (see Appendix 11.3) with adjustments made for product owner (PO), chief technology officer (CTO) and chief operating officer (COO) roles. These guides were used only to prompt conversation, follow-up and emergent questions were subsequently asked based on participant responses.

When designing the interview instrument care was taken to script open-ended questions to which there was no obvious or "correct" answer — for example, posing questions in the form of "in your opinion" and "in your own words". It was stated at the start of each interview that the focus would be on how the participant had personally experienced the evolution of agile at SoftwareCorps. It was also reiterated that there were no right or wrong answers to the questions asked.

For the phase two interviews, which were focused on exploring the influence of different leaders on the adoption of agile working, a new interview instrument was designed (see Appendix 11.3). The same approach to phase one was taken, where open-ended questions were designed to prompt conversation rather than explicitly drive the interview.

When reviewing the literature to explore what was known about leadership and agile working, the researcher came across transformational-transactional leadership theory and subsequently, the Multifactor Leadership Questionnaire (MLQ) (Avolio & Bass, 2004). The MLQ is typically used to measure leadership style and effectiveness quantitatively. However, for this study, it was used as a foundation for a more in-depth exploration of leadership. As such, closed questions from the survey were adapted to be open-ended so they could be used to guide semi-structured interviews. This adaptation allowed for a more in-depth exploration of leadership than the original survey format, exploring how and why leadership affected agile working. An example of how questions were adapted is provided below.

Example statements from MLQ (Avolio & Bass, 2004) which contribute towards a score for idealised influence, a facet of transformational leadership style;

“Talks about his/her most important values and beliefs”

“Considers the moral and ethical consequences of decisions”

Example of a question from the interview instrument used in this study, inspired by these statements;

In your experience, does (INSERT NAME OF LEADER) have any values or beliefs that are important to him?

* If so, what were/are these? (e.g. integrity? trustworthy? honest and open? moral? ethical?)
* Does he practice what he preaches? If so, can you give any examples? (e.g. makes decisions based on values and beliefs?)
* Do (INSERT NAME OF LEADER)’s values affect how you approach agile software development? If so, how?

The full leadership interview instrument can be found in the appendix. Although the MLQ could not be included in the appendix due to copyright restrictions, it can be accessed at www.mindgarden.com.

#### Participant Observation

The period of participant observation for this study lasted three-and-a-half-years, from January 2014 to August 2017. During this time, a variety of tools were used to collect data, including contemporaneous note-taking, photographs, copies of electronic communications and email, and other organisational documents. The approach to dealing with the quantity of data collected through an extended period of participant observation such as this was to categorise, index, and reflect on this data iteratively and incrementally (Myers, 2013, p. 144).

The decision to end participant observation was based on the researcher feeling no new insights were being discovered, combined with the sense that the original research questions could be answered using what had already been discovered (Myers, 2013, p. 143).

#### Data Collection Log

Given the volume and variety of data involved in this study, it was essential to establish a system for organising and managing everything from interview bookings to documents. Two tools were used for this; Excel and NVivo. Excel was used to track interviews, from the selection of the participants through to transcription and coding of the interview data. NVivo was used to store data such as transcripts, documents, and demographic information. It was also used to carry out coding and analysis as will be described later in this chapter. Copies of transcripts, audio recordings, documents, and photographs were also stored on a password-protected hard drive.

### Validity and Triangulation

In order to mitigate the weaknesses associated with interviews, such as bias due to poorly articulated questions, response bias, and inaccuracies due to poor recall and reflexivity (Yin, 2014, p. 106), multiple sources of data were combined. These included documents and participant observations. During interviews, the participants were asked to provide examples of what they were describing. This allowed the researcher to cross-reference examples provided with other accounts, contemporaneous notes of events, and organisational documentation. As suggested by a number of experienced qualitative researchers including Yin (2014, p. 102), an electronic tool (NVivo) was used to create a database and chain of evidence of all data collected. This tool was also used to code and analyse data, as will be discussed in Section 4.6.

As detailed in Section 4.5.4, a cross-section of participants were selected for interview, representing ‘various voices’ (Myers & Newman, 2007). It was also considered necessary to include a minimum of 2 individuals from each role, for each phase, to reduce the risk of individual bias. Given the depth of each interview, a large sample size was not deemed necessary to gather a rich data set (Starks & Trinidad, 2007, p. 1374). However, no upper limit was set. Instead, interviews continued until the researcher was comfortable that no new significant themes were emerging from the data and as such saturation had been reached (Guest et al. 2006, p.65; Francis et al. 2010, p.1230).

Several precautions were taken to minimise bias, both on the part of the researcher and through the selection of interview participants. First, participant bias was addressed through the selection of at least two individuals from each role, and through interviewing a cross-section of people from different levels of the organisation (junior developers to executive managers). As noted by Myers (2009, p. 133), this not only aids in the representativeness of diverse views but also triangulates the data and addresses some forms of bias. Second, researcher bias was mitigated through the provision of full interview transcripts to all participants interviewed, along with encouragement to review, correct, and clarify what was presented. Additionally, the researcher reviewed early visualisations of the data with a number of participants, as well as sharing early thematic analysis for the purposes of sense checking and obtaining multiple perspectives.

## Data Analysis

Although the distinction is often made between data collection and data analysis, in qualitative research, the boundaries between these activities are blurred (Myers, 1997). The researcher's philosophy, perceptions, and perspective all affect the gathering of data. The researcher defines the questions being asked, which ultimately determine what will be discovered. As Myers (1997) states *"the analysis affects the data, and the data affect the analysis in significant ways".* Consequently, in addition to coding, the process of interviewing, transcribing, and writing up of qualitative findings are considered part of the data analysis process in this study. Each of these are discussed in the following sections.

### Transcription

The process of transcribing interviews is in and of itself a part of the data analysis process. It allows the researcher to immerse themselves in the data, familiarise themselves with it, and start to form ideas about topics and themes. For this reason, reading through transcripts before coding them is recommended by virtually all methodologies (Saldaña, 1987, p. 18). The process of transcription itself involves reducing the data. The researcher removes the text from its context, including the nonverbal cues experienced during the interview. In this way, the researcher is selecting what is ‘seen’ even at the transcription stage (Myers, 1997, p. 56).

Each interview was audio recorded with the permission of the participant, which allowed for transcription at a later date. Due to time constraints, as a result of the researcher working a full-time job in parallel to carrying out this research project, some interviews were transcribed by a professional transcription company. Where this was the case, the researcher reviewed each completed transcript while listening back to the interview recording to ensure accuracy, especially around company-specific acronyms and names.

### Coding

The approach to coding taken in this study was based on principles set out by Salanda (1987). Computer-assisted qualitative data analysis software (CAQDAS) NVivo was used to store, code, and analyse data.

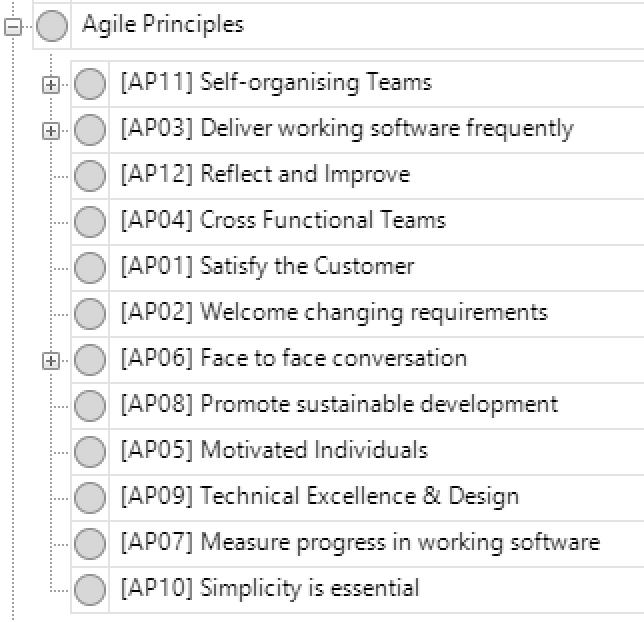
Salanda (1987, p. 3) defines a code in qualitative research as *“most often a word or short phrase that symbolically assigns a summative, salient, essence-capturing, and/or evocative attribute for a portion of language-based or visual data”.* Given the interpretive nature of coding, it is not a precise science. It is down to the coder to interpret and define codes to make sense of their data (Saldaña, 1987, p. 4). The purpose of coding is to reduce the vast amounts of qualitative data typical in this type of research to allow further analysis. This analysis includes identifying patterns and themes and enabling theory building (Saldaña, 1987, p. 8).

There are several styles of coding, the choice of which is often driven by the researchers underlying philosophy. For example, *In Vivo* coding involves using the participant’s own language to phrase codes. A *Descriptive* coder might attempt to capture and categorise a breadth of meaning, and a *Values* coder might attempt to label subjective opinions (Saldaña, 1987, p. 7). It is important to note that coding is not simply labelling but rather linking and assigning meaning (Saldaña, 1987, p. 8). It is through linking of these codes that sensemaking and theory building can take place. Coding is also a cyclical process; the researcher reflects, reviews, improves and iterates their coding through many coding cycles.

#### Coding Strategy

This section details the process of coding carried out in this study. Rather than waiting for all interviews to be transcribed before commencing coding, an iterative approach was taken. Batches of interviews which had been transcribed and reviewed were coded while transcription and data collection were ongoing. Commencing coding as data is being collected is the approach recommended by Salanda (1987, p. 17).

Both an inductive and deductive approach to coding was taken. A set of a-priori codes were inductively created based on the 12 agile principles, as shown in Figure 3. These were created to give some direction to the coding process, to help identify perceptions, and explore the manifestation of agile principles and practices in the day-to-day activities of teams. In addition to this, codes were deductively created based on the data. As such, the researcher's coding book evolved throughout the coding process.

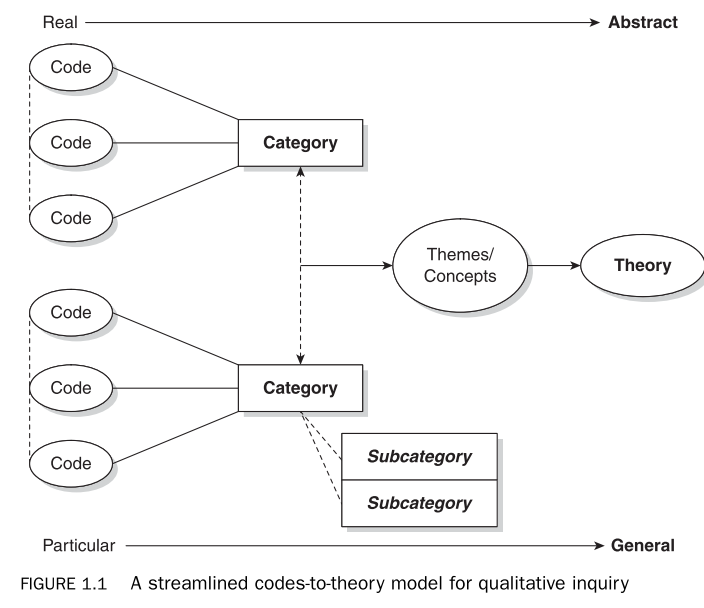
Figure 3: A Priori Codes

Throughout the coding process, numerous codes were created, revised, rephrased, added, removed, grouped, and categorised. Due to the iterative nature of coding, it was also necessary to revisit previously coded interviews as new codes and themes emerged. Keyword searches, a feature of NVivo, were used to aid in this process. An example of this is the code "seniority" which was created after several transcripts had already been coded. Due to immersion in the data, the researcher believed "seniority" appeared in earlier interviews but had not been coded under this label. The search functionality in NVivo allowed for targeted coding of references to this topic across multiple sources. Variations on the word "seniority" were used such as senior, manager, elite, rank. The context within which these terms were used was verified before coding. This verification ensured their meaning was consistent with the categorisation of the code.

It should be noted that although an auto-coding feature is available in NVivo, this was not used. Given the semi-structured nature of the interviews carried out, there was not enough consistency between interviews for this to be useful.

##### Codes-to-Theory

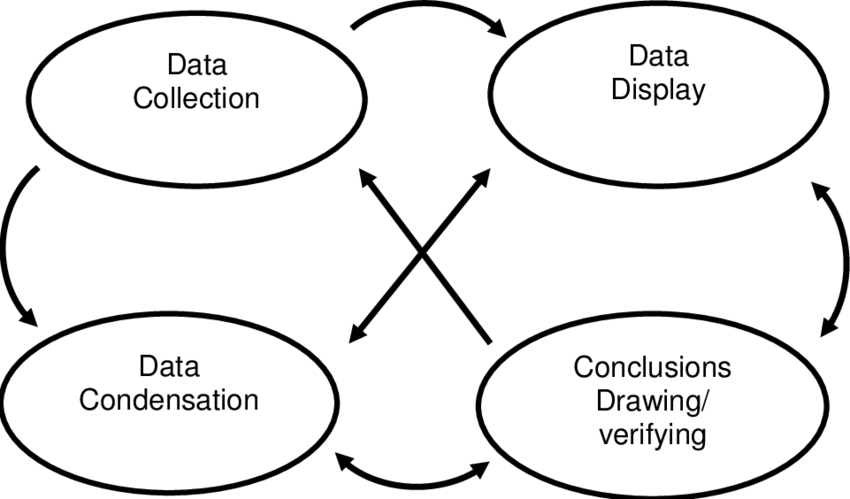
The process of recoding and recategorising (Saldaña, 1987, p. 10) was done continuously throughout the data analysis phase. However, once transcripts had been coded and keyword searches for newer codes had been carried out, this process was repeated to facilitate a codes-to-theory progression. The codes-to-theory model involves a process of taking lower-level codes defined by the researcher and organising them into categories and subcategories. This categorisation further reduces the data while maintaining the meaning behind it. From these categories, themes and concepts can be more easily identified and, ultimately, theories built. This process is visualised in Figure 4, taken from Salanda (1987, p. 12).

Figure 4: Codes-to-Theory Model taken from Salanda 1987, p.12

In the following section, the process of moving from coding to data visualisation is described.

### Data Display

Data analysis in this study was carried out according to Miles et al. (2014, p. 12) who define data analysis as three concurrent flows of activity; data condensation, data display, and conclusion drawing/verification. Data condensation includes note-taking, coding, and categorisation, as discussed in the previous section. In its original form, unreduced text in qualitative research is cumbersome and difficult to view as a whole and can overburden the researcher. Data display is the organised, compressed assembly of information to allow conclusion drawing, helping the researcher understand what is happening. Data displays include many types of matrices, graphs, charts, and networks, which are focused on presenting data in a way to enable viewing of a full data set in a single location. Conclusion drawing and verification is the interpretation of meaning from the data and data analysis, followed by testing for plausibility and sturdiness. These activities are carried out continuously and iteratively from a fluid and humanistic position, as illustrated in Figure 5 (Miles et al., 2014, p. 14). This section will focus on data display, conclusion drawing and verification.

Figure 5: Components of Data Analysis: Interactive Model (Miles et al. 2014 p.14)

Both matrices (with defined rows and columns) and networks (a series of nodes with lines and arrows between them) were used to display data in this study (see Appendix 11.7 & 11.8), as suggested by Miles et al. (2014, p. 107). These were created using a combination of coding, direct quotes and company documentation.

In order to search for themes and patterns in the data, a timeline view of the evolution of software development at SoftwareCorps was created. This timeline illustrated the "phases" of agile adoption, different approaches to software development, and significant punctuating events. Direct quotes were used as descriptors for each phase, along with other contextual information such changes to who held the role of CTO (see Appendix 11.6). From this visualisation, the theme of leadership began to emerge as each phase of agile evolution appeared to be linked to changes in leadership. Building on this early visualisation, and drawing on the literature (namely Cram & Newell, 2016) a network model of agile evolution at SoftwareCorps was created. This model helped to identify and categorise the different phases of agile working, linking this to Cram and Newell's agile fads research.

Matrices were used to collate data in a format from which comparisons could be made across the data. Interview transcripts in their raw form are sequential and linear and do not make for easy comparison or pattern identification (Miles et al., 2014, p. 108). This technique was also used when carrying out reviews of the literature to identify themes, as can be seen throughout this thesis (e.g. Table 6 and Table 9).

As suggested by Miles et al. (2014, p. 117), several 'tactics' were used to make inferences and draw conclusions from data displays created. These included scanning displays for anything that might 'jump out', noting patterns and themes, making comparisons, clustering and counting, consulting field notes, and written explanations of what was being inferred. This writing up is itself a form of data analysis, helping to clarify and reformulate ideas (Miles et al., 2014, p. 117). Data displays were also reviewed with a number of participants to sense-check and validate what the researcher had interpreted from the data. The literature was also revisited to determine if any other explanations could make sense of the data.

##### Emergence of the Leadership Theme

One of the most striking themes to emerge from phase one, through a process of data gathering, coding, analysis and reviewing the literature, was that of leadership. This process is illustrated in Figure 6. Many of the activities shown overlapped and were iterative, however, for simplicity the overall flow is presented as discreet stages.

A picture containing screenshot

Description automatically generatedFigure 6: Emergence of Leadership from Coding

During **phase one interviews**, participants were asked to recount their experiences of different ways of working during their time at SoftwareCorps, and many associated different attempts at agile working with changes in leadership (see Chapter 5 and 6 for more detail). The researcher **identified this as a theme** through conducting the interviews, and this was more formally identified during early coding. Ultimately this theme was confirmed through further data analysis.

Codes relating to leaders appeared in various forms in the coding book. An anonymised example of this is as follows;

* Timeline of SoftwareCorps
  + Nelson CTO
  + Julius CTO
  + Jamie CTO
    - Nelson Consultant
  + Zane & Andrii
  + ‘Phases’ generally
  + Early SoftwareCorps
    - Will
    - Early Development Approach
  + First Buy-out
  + Second Buy-out

As shown, codes relating to the ‘timeline’ of SoftwareCorps were created, with a node for each leader who held the role of CTO and other key events such as company buy-outs. These codes were used to create **early data displays** (see section 4.6.3) which further elucidated the theme of leaders as significant in the agile journey at SoftwareCorps. Given this, the researcher **returned to the literature** to search for theories which might explain what was being observed at SoftwareCorps, with a focus on leadership and agile adoption.

As was described in Section 3.4, many leadership theories were reviewed. However, that of transformational and transactional leadership appeared to be the strongest in terms of helping to explain what was being observed at SoftwareCorps. Consequently, some additional a priori codes were added to the code book which covered facets of transformational and transactional leadership theory.

A process of **coding and re-coding** was then carried out, as well as a **second phase of interviews** targeted at capturing data specifically around leaders and leadership. As described in Section 4.5.6, interview protocols were designed using leadership theory as a grounding. Through the process of coding and re-coding, codes were categorised to create a hierarchy and structure within the coding book. For example codes such as ‘dictate’, ‘controlling’, ‘micro-manage’ were mapped to facets of transformational leadership, and codes such as ‘empower’ ‘motivate’ ‘inspire’ were mapped to facets of transformational leadership.

### Write-up

The process of writing up the findings of this study, as presented in Chapter 6 and Chapter 7, also formed part of the data analysis process. In the process of organising thoughts, ideas and themes into coherent written findings, ideas were developed, matured and clarified.

## Summary

This study was carried out in two phases, each designed around a research question seeking to inform the overall research objective; to understand how the adoption of agile manifests in a real-life context.

The researcher’s interpretive perspective was appropriate for this study given the socio-technical nature of software development and the focus of the Agile Manifesto on values and principles. A qualitative in-depth case study approach was selected, driven by the exploratory nature of the research questions. The case study approach allowed the researcher to gain an in-depth understanding of the phenomena, complementing both the interpretive philosophical perspective and the objectives of the study.

The selection of the case was purposeful, based on the access the researcher had to the organisation and the commencement of an ‘agile transformation’ at the time of the study.

Various case study data collection methods were employed during three-and-a-half-years of fieldwork; semi-structured interview, participant observation, and documents. Interview was the primary source of data with observation and documents used to supplement the data. Coding of the data was carried out according to Salanda (1987), and the creation of data displays according to Miles et al. (2014).

This chapter has presented the research methodology and methods used to collect and analyse data. Subsequent chapters will present the findings of this study. First, a history of SoftwareCorps' agile journey will be presented, which provides context for the research findings.

# A History of SoftwareCorps and their Agile Journey

The purpose of this section is to provide some background of the case under study, detailing key events in the organisation’s history in order to provide context within which the research project took place. This chapter begins by providing a summary of the participants and their roles for the purposes of reference before detailing the agile journey at SoftwareCorps, from its founding in 1993 through to the period during which the researcher carried out the study, ending in 2017.

## Summary of the Participants and Key Roles

The following table (Table 14) is provided as a reference for the reader. It presents a summary of all interview participants and their roles, as well as some key individuals who were named during interviews (such as legacy managers) but who were not interviewed. Pseudonyms are used in all cases to provide anonymity, as was agreed during the interview process.

Table 14: Summary of the Participants and their Roles

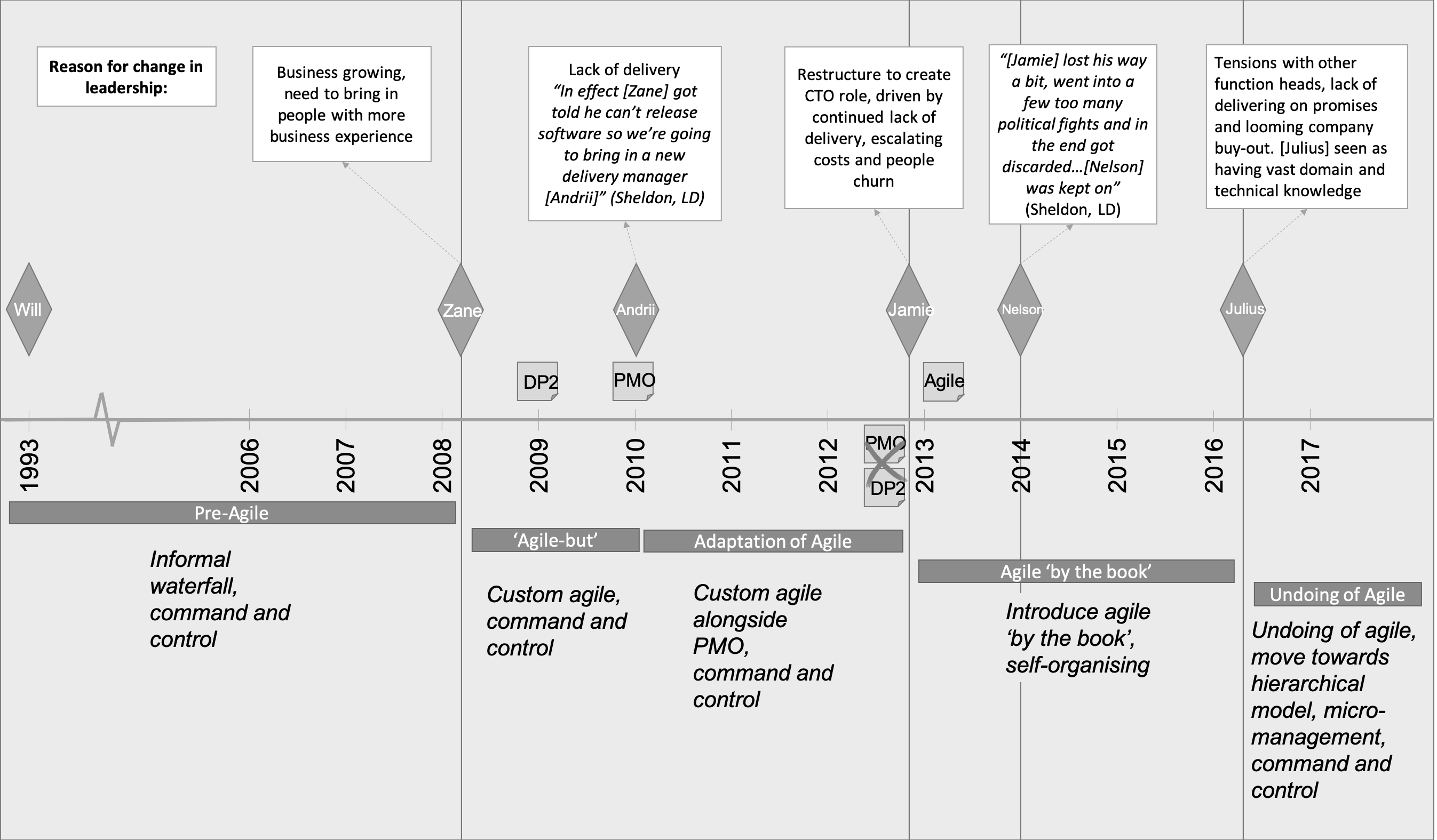
|  |  |  |  |
| --- | --- | --- | --- |
| Group | Role | Participant | # of interviews (notes) |
| Executive Management Team | CTO | Nelson | 2 |
| Julius | 2 |
| Jamie | n/a |
| C-level | Richie | 2 |
| Will | n/a |
| Executive level | Zane | n/a |
| Andrii | n/a |
| Chad | n/a |
| Kai | n/a |
| Software Development Leadership Team | Lead Developer (LD) | Woody | 2 |
| Ted | 2 |
| Reece | 2 |
| Drew | 1 (P1 only) |
| Sheldon | 2 |
| Emmett | 2 |
| Laurie | 2 |
| Don | n/a |
| Scrum Master (SM) | Adrian | 2 |
| Rowan | 1 (left company) |
| Frankie | 2 |
| Paul | 1 (left company) |
| Software Development Team | Developer (Dev) | Quinn | 2 |
| Luke | 2 |
| Michael | 2 |
| Bart | 2 |
| Darren | 1 (P2 only) |
| Gabor | 1 (P2 only) |
| Darius | 1 (P1 only) |
| Developer in test  (DIT) | Lance | 2 |
| Taylor | 1 1 (P2 only) |
| Gabriel | 1 |
| Product Team | Product Owner  (PO) | Blair | 2 |
| Joseph | 1 |
| Harper | 1 |
| Will | 1 |
| David | 2 |

## Agile Journey at SoftwareCorps

The working practices adopted within an organisation can be complex and nuanced, and as organisations and the environments they exist within change so must the way they operate.

SoftwareCorps was founded in 1993, starting out as an IT consultancy and later developing what would become a leading platform in its market and evolving into a multi-million-pound global organisation. Evidently working practices appropriate for a start-up with less than 15 people would not be fit for purpose for an established business of almost 200 employees. As you might expect, working practices evolved and grew with the business over time and this evolution was punctuated by specific periods of change and key events.

Figure 7 illustrates the different periods of evolving working practices at SoftwareCorps, along with reasons for leadership changes from the founding of the company through to 2017. Terms given to the overall working practices associated with each period are also presented; Pre-agile, Agile-but, Adaptation of Agile, Agile ‘by the book’ and Undoing of Agile. In the following sections SoftwareCorps’ agile journey is presented in detail, structured in terms of the approach taken towards agile during specific periods. Each approach is then defined at the end of each section along with a summary of the difficulties encountered.

Figure 7: Timeline of Agile journey at SoftwareCorps

### Pre-agile (1993-2008)

In the early years at SoftwareCorps, when its founder Will was still leading the business, there was no formal software development approach to speak of. The company was small enough that Will could be very hands-on with his employees and manage delivery from the top. Employees were acutely aware that as the founder of the business, Will had a targeted interest in ensuring the profitability of the company. Will therefore made all key decisions and reportedly directed his colleagues clearly in what was expected of them and where he wanted to take the business. At this time the company did not officially follow any particular methodology. However, based on descriptions by the participants of how they worked, their software development approach at that time is best described as an informal waterfall model following a staged approach.

*“Back then there was no real software development approach as such. You did what you were asked and that was “Right then, this client needs this thing. Here are the details of what they need.” It was all very spec-based…it was a staged delivery system” (Sheldon, LD)*

The development and testing departments were separate, forming distinct teams, with the standard lifecycle following a design, build, test, and release process, though this was never formally defined. Technical assessment documents were created upfront, proposing how the software would work, which required signoff from the entire development department before the build phase began. The build phase was followed by a code freeze and testing cycle which lasted up to two months. A project would pass through each phase taking a typical waterfall route, if a project failed the testing phase it would return to development and the testing process would be repeated again once the features were passed back from the developers to the testers. Every project had to pass a phase of ‘stabilisation’ in which no more code was written and no serious bugs were found, if concerns were raised the stabilisation ‘clock’ would restart until the testing team were happy to sign off the release. This often led to discord between the development and testing teams as responsibility for quality laid with the testers, whereas developers were keen for new features to be released.

This spec-based approach was perceived as suitable for SoftwareCorps at that time as they worked predominantly on bespoke products for specific clients on a project by project basis. Projects they tendered for would be designed upfront, with timescales and costs agreed and signed off before development started, in line with the customer’s request for quote (RFQ) processes and often in competition with far more established service providers. As the company grew they moved away from purely bespoke work towards a more product-based approach centred around their core platform.

Releases of the core platform were events which occurred every 6 months at a pre-defined date with the contents of those releases agreed up-front in the form of the company roadmap, created by Will and his product manager Chad. Under this model features would need to ‘make the release’ or risk being delayed for 6 months until the next one was scheduled. A consequence of this was a desire to get “just one more thing” into a release, which often lead to delays and even larger releases. Most notably, at the point releases were rolled out there were reportedly situations where the requirements no longer satisfied the needs of the client;

*“We ended up spending £1m on a product that no one really seemed to want in the end, which I guess is a lot to do with not releasing very often and being still fairly waterfall.” (Quinn, Dev)*

To summarise, this ‘pre-agile’ period denotes the period in SoftwareCorps’ journey where a predominantly waterfall approach to software development was followed, evident from the staged lifecycle and separation of development and testing. During this period projects were often large and complex, and in some cases took years to deliver, resulting in reduced impact of features when they eventually hit the market. There was also an unhealthy relationship between each department (product, developers and testers) and lengthy periods of bug fixing reportedly resulting from a lack of collaboration and lengthy feedback loops between coding and testing.

### Agile-but (2008-2010)

In the years that followed the business went through a period of rapid growth and in 2008 it was purchased by a large firm. More executive managers were introduced to support the growing business, which for the software development department meant the introduction of a ‘head of technology’ position filled by Zane. It was Zane who first introduced the language of agile into the company, sending selected individuals for external training and inviting a consultancy firm to build a process based on agile but bespoke for SoftwareCorps. Zane held the opinion that off the shelf solutions such as Scrum may work for other organisations but that SoftwareCorps would need something more bespoke;

*“[Zane showed me] some of his ideas and what he’d planned to do and it was all about bringing in agile and it was all about getting the consultant in…[he said] ‘I know there is Scrum out there in the world… [but] we do things the way it works for [SoftwareCorps] so let’s customise it’” (Sheldon, LD)*

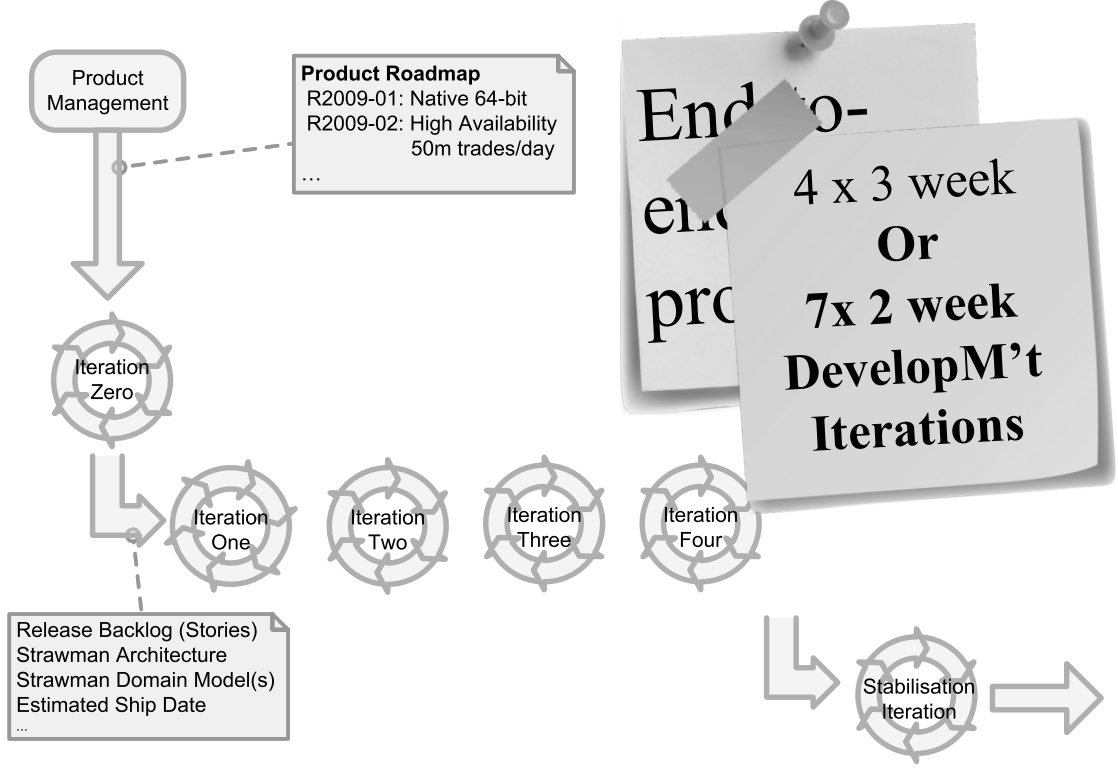
The consultants were engaged to design a process which would consolidate learnings and experience from past SoftwareCorps projects with the best of what Scrum and lean had to offer. With input from a small number of developers selected by Zane, the consultants created ‘Development Process v2’ (DP2). This process was specifically designed as SoftwareCorps’ own brand of agile, heavily customised to suit the organisation. Although there was an awareness in the software development teams that agile methodologies existed, they had little if any direct experience of it.

*“I think we fell into the trap that a lot of places do which is to immediately decide that, you know, in principle [agile] was good but in order to make it work here we needed to change it. Without actually having tried it first. So they came up with this [DP2]” (Woody, LD)*

In December 2008 the new process was announced and teams were trained through group inductions. On completion of their engagement the consultants provided a large document detailing the process they had created, this document became affectionately known as the ‘DP2 Bible’, in part due to its considerable size.

The process was based around an iterative, staged approach. A product roadmap was defined up front followed by an ‘iteration zero’ where stories were created, architecture and domain models designed and shipping dates estimated. Four 3 week or seven 2 week ‘development’ iterations then followed and finally a stabilisation iteration for final code freeze and testing would be completed before a release was signed off. Much of the design, estimation and planning was done up-front with project delivery staged over several iterations, however a client-facing release with working software was only available after stabilisation was complete. Figure 8 illustrates this workflow and was taken from a presentation used to train people in the process.

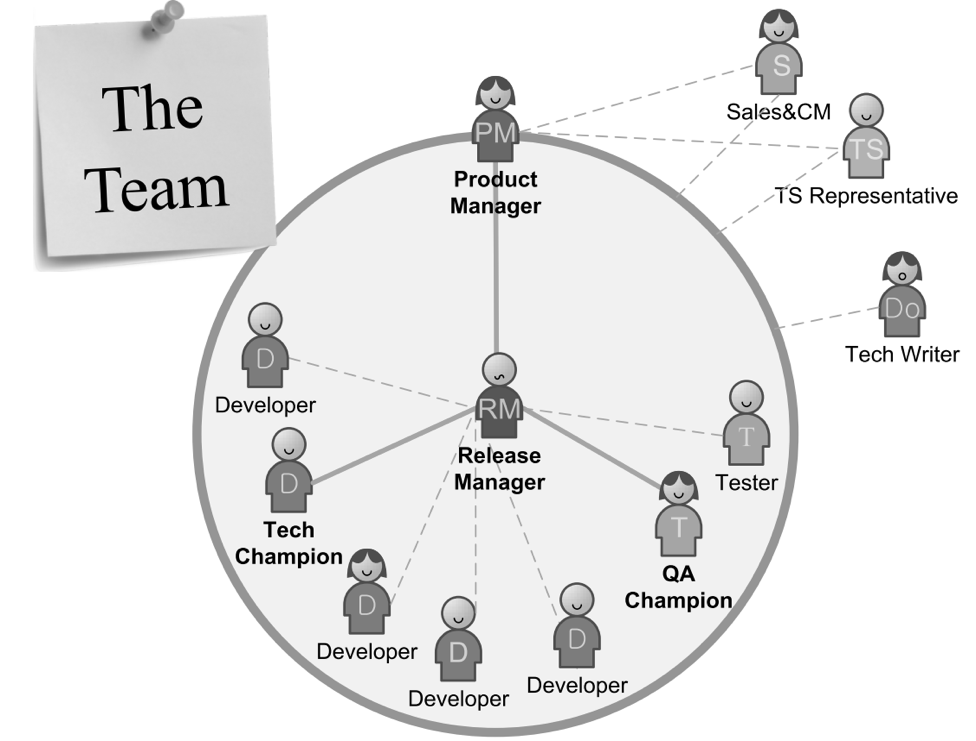
Figure 8: Development Process v2 (DP2)



The predominant differences between the new and previous process was the language used, the types of measurements introduced and the insertion of so-called ‘information radiators’ (whiteboards) to increase visibility of development. Requirements were now provided in the form of stories, not specs, and the teams would be required to break down the project into iterative blocks. They were also asked to commit to dates for each milestone and to provide reports on team burndown and bugs to management in an attempt to improve visibility of progress.

There were four roles created as part of DP2 and individuals performing these roles formed part of the wider team of developers and testers on a project, as illustrated in Figure 9. The roles were; a QA champion who was responsible for the ‘functional and non-functional quality’ of the release, a tech champion who was responsible for the ‘technical quality’ of the release, a product manager who ensured the product roadmap ‘added value for clients’ and a release manager who was responsible for balancing the views of the other three roles in order to ensure ‘timely delivery’. This role was described as being to DP2 what the scrum master was to Scrum *“with the main difference being it was a part-time role…fulfilled by a senior member of the team”* (Adrian, SM). This added responsibility was seen as a form of promotion and therefore a desirable role to achieve.

A release manager was given ultimate responsibility as to whether a release would go out, often putting them at odds with other members of the team. It was the release manager’s responsibility to enforce DP2, schedule meetings, maintain the velocity of the team, manage risks, ensure delivery was on time and of high quality, and to do resource allocation. It more closely resembled project management than the role of a scrum master (which usually holds no formal authority over the team and focuses on coaching and facilitating).

Figure 9: Software Development Teams

The motivations for introducing an agile way of working were reportedly unclear to the software development teams, with speculation ranging from ‘a way to micromanage’ to ‘management wanting to try the new and exciting fad’. The new process left SoftwareCorps with a software delivery lifecycle which neither provided results nor control over what was being delivered;

*“The first time we tried to go agile – and I say tried because I don’t think it was a big success – was when [Zane] joined us as our head of technology and it was he who devised the first agile process…my recollection of that period is that it was just utter chaos. I mean there was no real control. So we had the paraphernalia associated with agile development in the sense that there were story cards and boards and stuff, but it wasn’t really agile, it was just chaos.” (Harper, PO)*

Based on the participants’ own accounts, there was very little understanding in the development department of what agile was really about. The business was told that methodologies such as Scrum were reported to improve productivity and the software development teams had been introduced to ideas such as stories, scrum boards and velocity. In reality however, many of the central tenants of agile working appear to have been missing. Product increments were not completed within each sprint, releases were not happening *“anywhere near a sprint by sprint basis”* and *“retrospectives were rare”* (Quinn, Dev).

Whilst reflecting on this period, many participants noted that they would not consider how they worked then as agile, knowing what they knew of agile at the time of the interviews for this study. Notably, participants felt that simply redefining the development process was not enough, and that by taking parts of a recognised methodology such as Scrum and manipulating it to fit with existing working practices SoftwareCorps had missed the point;

“*So rather than look at the core values or the core ethos of agile, everyone just probably thought Scrum and thought “How can we adapt this to our means?” and then actually I think we threw away a lot of the bits that actually made Scrum agile and were just left with some weird process” (Blair, PO)*

In summary, this period of SoftwareCorps’ journey is labelled as ‘agile-but’, denoting an approach where some elements of agile methods are used whilst much of the underlying values, such as releasing regularly, are omitted. Working in this way in essence meant that the language and paraphernalia of agile working practices were evident, but on closer inspection little had changed in the way software was delivered. A staged approach was still in place with a period of stabilisation for code freeze and bug fixing, and a separation of responsibility between technology (coding), quality (testing) and delivery (releasing) continued to cause tensions. Products would take many months to be delivered and customer-facing releases happened only at the end of the project. To some, this was evidence that ‘agile’ as they had come to know it simply didn’t work.

### Adaptation of Agile (2010-2013)

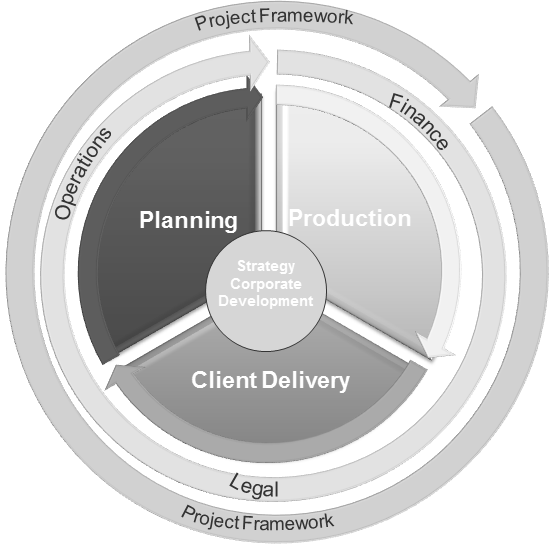
When put to the test issues with DP2 began to arise. Despite the changes to the process of software development, the type of work and projects it was applied to remained the same. Large and complex projects, although now broken down into iterations could still last many months before a release was available for the customer. The promise that a new agile process would solve delivery problems for the business had not come to fruition, other executive managers in the business were unhappy with a lack of visibility and control and this led to the introduction of a delivery manager into the software development department.

*“The problem is we then put it into practice on some big projects… ‘Juggernaut’ and things like that which unfortunately kept its name…we just weren’t in a place for [agile] to work. But [Zane] got the ramifications of that…in effect he got told he can’t release software so we’re going to bring in a new delivery manager, and that’s when [Andrii] came on.” (Sheldon, LD)*

Andrii first joined the company as a production manager brought in to introduce better governance and control of software projects. He had a background in project and programme management as well as software development, and shortly after he started SoftwareCorps embarked on the introduction of a Project Management Office (PMO) and a project management framework based on Prince 2. This PMO team created the ‘project framework’ which was established to address what executive management saw as key deficiencies; management and allocation of resources, delivery of commercially successful projects, and planning.

*"SoftwareCorps does not consistently deliver new projects on time or on budget. This has been mainly due to lack of qualified Project managers and a supported framework to do so... Senior management does not have a good handle on whether resources are used optimally and most effectively. If the company is to grow and prosper, more effective management of resources and better delivery of commitments is essential." (Delivery Project, Project Initiation Document – Business Rationale)*

The project framework was designed to sit around the existing software development process (see Figure 10 below) and allow executive management more control and oversight of what was going on in the development department, known at this time as ‘Production’. The PMO was established as a governance framework with a focus on formalising project plans, budgets and scope changes, as well as providing reports and metrics to senior management. The vision was to establish a PMO department who would police and administrate the process to ensure it was followed and maintained.

Figure 10: Project Framework

For each project a ‘Project Initiation Document (PID)’ was required, which needed to be signed off by executive management before development commenced. This would include high-level details of requirements, and a budget and milestones which were monitored by a steering committee through pre-defined ‘gate reviews’ and weekly reports. Any deviations from this plan required a formal change request to be raised and approved by the steering committee. This project process followed Prince 2 principles and was largely orthogonal to the software development process (indicated in Figure 10 as “Production”) by design.

In 2010 the business underwent a structural reorganisation, moving from business units split by client type to departmental functions (Operations, Support, Client Management etc.). This led to Zane and Andrii being given new roles; Zane as head of planning, with tactical development, product and project management teams, and Andrii as head of production, with software development, QA and documentation teams. The two function heads would be required to work closely together in order to deliver projects, however the organisational structure now led to a discord between the two departments. With the production department housing those that built the software and the planning department those who designed and prioritised what should be built, the responsibilities and delineation of decision making was unclear;

*“the divides were all wrong again, so the people that did weren’t allowed to make the decisions on how they did it. The designs were put into ivory towers and dictated from on high. The teams lost the one thing that made them productive, the feeling of [being a] team” (Sheldon, LD)*

Participants described conflict at the top as commonplace, and this discord filtered down through the ranks. A divide formed between architects and other developers, conflated by disagreement between Will and Zane regarding who would lead software development teams and who had ultimate say regarding design and strategy. They resolved to have a tactical software development team who would advise teams on architecture reporting to Zane, and senior developers who managed the team day to day reporting to Andrii. Developers under Andrii saw the tactical software development team as ‘architecture astronauts’ who would drop-in to a team to do the stimulating and exciting work whilst they were left to deal with ‘business as usual’ day to day tasks;

*“[The planning department] was like an elitist department… architects got dropped onto all the teams. They had their own meetings about what they were doing and which way they were going…their role was really to say “this is how it's going to be done” and then everyone else executed on that…proper architecture astronauts. And that really was very divisive some people were extremely upset about that.” (Emmett, LD)*

The product team reporting into Zane also had difficult relationships with Andrii’s developers, which lead to problems articulating requirements and a lack of visibility of what was actually being developed. The perception was that prioritisation was not being done effectively, with teams deciding what to work on without consulting the product managers (who were not part of the software development team), and product managers who had ‘go-to’ developers they would use to circumvent the process and get their projects done. During this time there was reportedly a continued lack of delivery and significant unhappiness from the development department regarding the number of projects in flight and a lack of clear leadership regarding prioritisation.

This period in SoftwareCorps’ journey has been labelled ‘adaptation of agile’ as the organisation attempted to blend what it knew of agile with a structured project management framework based on Prince 2. In doing this, SoftwareCorps attempted to adapt agile to suit their existing organisational structures. During this period there were a number of difficulties, firstly a fundamental conflict between traditional Prince 2 project management and agile software development approaches. With no changes being made to the approach to software development the new project management process did not achieve its aim to improve delivery of projects ‘on time and in budget’, rather it introduced additional bureaucracy around getting software delivered in an attempt to increase control by senior management. Secondly, increased discord between the newly created ‘functions’ due to separation of decision-makers from those doing the work, and a lack of clarity around responsibilities in these areas.

### Introducing Agile by the book (2013-2014)

As consequence in part of the ongoing delivery problems SoftwareCorps was experiencing, in late 2012 significant changes were made to the executive management team with Zane and Andrii’s roles being made redundant and the role of chief technology officer (CTO) being created. It was the introduction of a new CTO, Jamie, which began the next period of change which included another restructure and the formation of a new ‘technology’ function encompassing all developers, architects, QA and documentation teams. The project management team were consolidated to report directly into Richie (COO), and product management was moved under a new head of Global Product Management.

Jamie began by running ‘townhall’ sessions to present his vision for SoftwareCorps, in which he set out his aspirations to create self-organising, cross-functional teams who would be empowered to decide how best to deliver their products. Part of his vision was to move towards a ‘self-organising’ management model, an entirely new concept at SoftwareCorps which differed dramatically to the existing command and control model. Participants described these early interactions with the new CTO as very positive and noted a difference in his approach when compared to his predecessors.

“[Jamie] seemed to…[want] to run a technology department and he's interested in having that run well with the rest of the company … he's not assuming the worst. He is assuming that there is good stuff there and it needs bringing out, or more of it needs bringing out. It's nice starting off positive rather than negative.” (Emmett, LD)

Jamie (CTO) engaged an Agile Coach (Nelson) on a short-term contract for 2-3 days per week to help him understand how the teams worked. Together they established that the current processes (DP2 & PMO) were neither agile nor working for the business. Although the existing process had been introduced as ‘agile’ there was no real understanding in the business of what this meant and in Nelson’s words, “not one iota of agile or Scrum is visible in this organisation”. Consequently, they kicked off an initiative called “Technology Transformation” and through this they retired both the project framework and DP2. Together they began training and coaching the business in the basics of agile as a methodology and mindset, advocating for doing agile “by the book” (Woody, LD).

Nelson began by running ‘introduction to agile’ sessions across the business, starting from the basics of what was meant by ‘agile’. After observing the teams for a few weeks and familiarising himself with the business, he asked for a team to volunteer for agile coaching. The volunteers had to agree to be coached for a minimum of 6 sprints (12 weeks), following the principles and practices of Scrum to the letter. The technology function had become an insular unit, difficult to penetrate or influence from the outside and as such the first test was to find a team of developers and QA who were willing to accept an external person coaching them. The approach taken was to create an environment where all teams had access to coaching, but would have to ‘self-organise’ and reach out for it. One by one each team was given a period of coaching and internal coaches were trained on the job, carrying out the role of scrum master (although this was not yet recognised as a full-time role).

The first software development team Nelson (CTO) engaged with were at odds with their product manager. Their interactions were aggressive and adversarial which caused a breakdown in communication, especially between the lead developer and product manager. There was a perception by the product team that the software development team were working too slowly and that what they were producing was of low quality. The software development team reportedly felt like they had to justify every design decision they made and that the product team weren’t providing them with the requirements they needed to deliver effectively. Over a number of months, the scrum master and Nelson (CTO) worked with both parties to establish a healthier working relationship. To do this they began by setting expectations of the roles, such as what a product owner did and didn’t have responsibility for, and helping the software development team to translate requirements into software which could be delivered incrementally. The example set in this team became the norm for each of the software development teams as they adopted the scrum methodology, with product owners becoming embedded in the teams.

Whilst Nelson worked on the ground, at the top Jamie (CTO) was struggling to get traction for his vision. He believed that in order to create fully functioning agile teams he needed them to be cross-functional, and to achieve this he wanted the delivery function (which housed client-facing technical support and internal IT) to report into him. This was seen as a power grab by others in the business and led to a series of political battles at management level which appeared to ultimately cause to a loss of faith in Jamie’s (CTO) ability to deliver on the promises he had made. Jamie (CTO) stepped down just over a year after he joined SoftwareCorps and Nelson was asked to step into the CTO role temporarily to smooth the transition.

*“[Jamie] lost his way a bit, went into a few too many political fights and in the end got discarded.” (Sheldon, LD)*

This period in SoftwareCorps’ journey is labelled ‘introducing agile by the book’, a term used by participants when describing this period. It was this point in time that participants associated with beginning to build an understanding of what agile meant, starting with getting back to basics and following agile processes initially ‘by the book’ rather than adapting them. There were a number of difficulties to overcome during this period; Jamie (CTO) had to convince the wider organisation that in order to adopt a truly agile approach, there would need to be significant changes to the organisation as a whole, not just to the technology department. In addition to this, significant efforts had to be made to coach teams and managers who had misinterpreted agile practices and learned ‘bad habits’, as well as coaching the wider organisation in new concepts such as self-organisation.

### Embedding Agile by the book (2014-2016)

Through his engagement as an agile coach, Nelson (CTO) had worked closely with a number of individuals to train them up as internal scrum masters with the intention of establishing an agile community of practice which would continue to develop and coach SoftwareCorps’ agile capability when his contract ended. He had also worked closely with many of the software development teams to help them understand and adopt Scrum, and engaged with the wider business to share his knowledge and experience of agile working, in particular the need for an organisational perspective on agile delivery. Through his work with the software development teams, he was also able to work closely with the product team.

During his interview, Nelson (CTO) explained that his decision to stay on as interim CTO was motivated by a desire to nurture the changes he had been working with Jamie (CTO) to embed. He recalled that Richie (COO) had suggested to him that *“everything you’ve done will go [if you leave], because it’s in its infancy. It is working, but it’s not working”* and that if he were to accept the role as interim CTO Nelson would have *“more authority to do what you have been doing”.* This show of support was a pivotal moment;

*“I needed someone to give me the cover at the senior management, [to empower] me to do it. If you took the first one away [management support] I couldn’t have done anything. So I accepted with both at the same time…what I’d done could not have survived [otherwise].” (Nelson, CTO)*

The announcement that Jamie (CTO) had left the company and that Nelson would be interim CTO happened abruptly. On Friday Jamie (CTO) had been carrying out his role as normal and by Monday morning the technology department were informed of his departure. Based on the role Nelson (CTO) had carried out as an agile coach, many of those who had worked with him assumed he would continue coaching the software development teams even in his role as CTO, however with the new role came pressures and responsibilities which meant this was not possible long term.

One of the issues Nelson had identified during his time working as an agile coach was that there were a number of people in the technology department who were working against the move towards an agile way of working - these individuals had long tenure with the organisation but were not on board with the change to their working practices. In his new position as CTO Nelson now had the authority to deal with this issue. He acted where he believed Jamie had not and a number of long-serving employees were asked to leave.

“The most regrettable thing that I do, I feel like assassin most of the time, that I had to identify whom will never be here. There is no way that they will learn… there is no way they can transition to a new way of working…I believe one of the failures of [Jamie] was that I gave him the names that “these people are not technically, or in terms of knowledge of how to do work, they will not be with us” and he retained them. All of the people that I [then later] had to part with.” (Nelson, CTO)

Although those working closely with Nelson understood his motivations it was a divisive action which caused ripples throughout the department.As Nelson continued in the role of CTO, no longer purely an interim role, the lack of technical domain experience was a concern for some, and there was a perception that his focus was too much on the agile process and not enough on technology for someone in the CTO role. There were concerns that by focusing on the agile process the technology itself was suffering, becoming old and in need of renewal.

During their interviews, participants discussed a variety of significant changes which they attributed to Nelson (CTO). During his tenure Nelson was reportedly instrumental in establishing communities of practice with the lead developers and scrum masters, which were empowered to make decisions and implement change. He also gave more autonomy to teams to organise their work and supported the formalisation of the scrum master as a full-time role. Outside of his work with the teams, a focus for Nelson was establishing agility as an organisational concern, not just something for the software department. Through his own research and experience in implementing agile practices, he was an advocate of breaking down organisational boundaries and building cross-functional, goal-driven teams. There was however a perception that over time Nelson’s approach changed with Adrian (SM) recalling that Nelson (CTO) was very positive when he joined SoftwareCorps, but that over time he appeared to get ‘worn down’ by events, and participants held varying opinions on whether the changes he introduced were ultimately for the better.

*“It’s interesting in hindsight as well people’s opinions because of all the sort of regimes, for want of a better term, I think his has been probably the most divisive. Some people do not look at that time fondly, but usually when you drill into it there’s kind of times within the time, it’s not just the whole thing, but people’s experiences do vary and people maybe didn’t see why he was doing things.” (Adrian, SM)*

Nelson (CTO) developed a number of enemies during his time as CTO due to some of his divisive ideas and actions (such as redundancies and restructuring his teams) and Richie (COO) noted that although during his earlier years at SoftwareCorps Nelson (CTO) had been successful at energising people, over time the benefits of this dwindled and Richie (COO) suggested that Nelson (CTO) had become defensive over what he had achieved;

*“I think what we probably didn’t realise is somebody who’s good for 6 months but as the organisation learns what he has to teach his value diminishes… then you’re not only fighting the business decisions that need to be made, you’re watching the destruction of your own legacy” (Richie, COO)*

Richie (COO) also commented that he believed the CEO Chris had built tensions into the organisation by design, which kept leaders within the business in competition with each other and created a highly charged political environment where power struggles and shifting alliances were a regular occurrence. It came as no surprise therefore that as a result of SoftwareCorps’ acquisition by a large conglomerate (Tech Inc.) in 2016 these power struggles resulted in both the CEO Chris, and Nelson (CTO) leaving the company. When describing the events that took place leading up to Nelson’s (CTO) resignation, Richie (COO) commented that a number of people had pushed him to *“shove Nelson (CTO) out the door”* before his resignation and Richie (CEO) reflected that he felt partially responsible for Nelson’s downfall;

*“[Nelson (CTO) was] not the genius god almighty we thought we had. [He’s] just like the rest of us and I think we built [Nelson] up and then we knocked [him] down… you have a good 2 or 3 years and all of a sudden you are rewarded with more responsibility…but higher expectations and then life happens, events occur and then you become defensive about things in a way that’s just not healthy” (Richie, COO)*

Through a combination of SoftwareCorps’ acquisition by Tech Inc. who were not advocates of agile working and a breakdown in relationships between Nelson (CTO) and the wider executive management team, Nelson (CTO) resigned and was replaced in his role by Julius in early 2016.

This period in SoftwareCorps’ journey is labelled ‘embedding agile by the book’ as it depicts the time in which Nelson worked to embed practices introduce by his colleague and predecessor Jamie (CTO). Many participants associated Nelson (CTO) with establishing agile as more than just a process, with a focus on changing behaviours and addressing wider organisational blockers to agile working. This focus resulted in a number of redundancies and restructuring of departments which divided opinion at all levels of the organisation, and over his tenure participants described a decline in Nelson’s success, which some suggested accompanied the change in his role from agile coach to CTO. The executive management team began to doubt Nelson’s ability to fulfil the role of CTO and with an increasingly political environment and significant pressure due to SoftwareCorps’ acquisition by Tech Inc., Nelson left SoftwareCorps.

### Return to Command and Control (2016-2017)

According to a number of participants, including Sheldon (LD), a key motivation for selecting Julius to replace Nelson as CTO was a perception that his approach would be more closely aligned with how Tech Inc. operated, with a focus on deadlines and delivery. Participants commented on their expectations that SoftwareCorps would be required to adopt their new owners’ existing policies and procedures, and highlighted the uncertainty surrounding job security and the direction of the business. Richie (COO), who in his role was closest to the detail of the acquisition, articulated his expectation that certain parts of the business, such as marketing and legal, would be centralised and integrated into existing operations within Tech Inc. and that SoftwareCorps would exist *“more like a brand than a business”*, consequently he anticipated many of SoftwareCorps’ roles would change or become redundant. The uncertainty around when and how changes would be made was drawn out for over a year as the details of the acquisition were negotiated, creating an environment of insecurity.

Of particular note by a number of participants were concerns that Tech Inc. were applying significant pressure on SoftwareCorps, and that there would be serious repercussions if SoftwareCorps failed to deliver results. In order to meet Tech Inc.’s expectations SoftwareCorps increased the number of projects in-flight which required a re-organisation of the technology department. Julius (CTO) presented the changes he would be introducing to support this in a technology all-hands meeting. Figure 11 is taken from his PowerPoint presentation;

Figure 11: Organisational changes introduced by Julius

**ORGANIZATIONAL CHANGES**

* Technology will be organized into four competency clusters
* Each cluster will be responsible for a set of products
* Teams in each cluster will have separate planning sessions, stand-ups and retrospectives
* **Lead developers and scrum masters will be responsible for organization, synchronization and distribution of work within their cluster** e.g.
  + There may be more than two teams at some point in time
  + Lead developers will attend all of the planning and design sessions

Things that won’t change:

* Scrum
* Sprint demo
* Each cluster will agree the definition of done for their products

There were significant changes imposed on the lead developers and scrum masters in this re-organisation, in particular to the way Julius (CTO) expected them to manage projects and teams. Julius (CTO) was an advocate of Rational Unified Process (RUP) (an iterative software development framework with four stages – Inception, Elaboration, Construction and Transition) and Julius set his expectations that lead developers would use RUP principles.

In particular Julius (CTO) emphasised that during the ‘inception’ phase of a project the minimum number of resources should be allocated, and that usually this should be the lead developer as they were the most skilled. Once a project reached the ‘elaboration’ and ‘construction’ phases where the most effort was required, additional resources should be added before reducing resources again during the ‘transition’ phase. He also explained that projects should overlap to ensure the most efficient use of resources, such that during the transition phase of one project, the inception phase of the next should begin. A particular point of contention for Julius was what he saw as ‘over-resourcing’ of projects;

*“I was giving the team lead some advice around a very simple process model for him that he should apply and I was telling him at what point in time he should use how much resources and I was explaining to him why before a certain process step he shouldn’t put too many people on it. Then he was behind the deadline and of course he put too many people on it and of course it made things worse…he realised himself that he’d [messed] up there.” (Julius, CTO)*

Another change to the day-to-day operations for the lead developers was a significant reduction in the number of meetings they were asked to attend, which was seen as a positive, however this also meant that they were less involved in the decision-making process and their freedom to self-organise was subsequently reduced;

*“[Julius has] done one thing – zero meetings unless there’s an absolute reason for a meeting…there’s two flip sides to this particular equation. (1) Yes, people now have more time to hands-on code, though (2) they don’t have access to the information they necessarily had before and there are decisions now being made for the group that the group may have made before…there have been quite a few specific dictatorial decisions made already” (Sheldon, LD)*

When describing the period in which Julius was CTO, many participants spoke of a move away from agile working and a decay in their agile practices, such as collaboration and communities of practice. Frankie (SM) recalled how he had feared that agile working would *“all fall apart”* as a result of Julius’ transition into the role of CTO, having worked with him in his role as a lead developer, and that it was sad to watch something he had been part of building deteriorate;

*“[Julius’] ideas of how to do things are, from my point of view, not scrum. They’re waterfall or they’re manager top-down ideas. It’s basically not self-organising at all… technology isn’t working in an agile [way]...The only thing that’s agile about our technology work at the moment is the fact that we’ve got boards up.” (Frankie, SM)*

Midway through Julius’ first year as CTO, he restructured the scrum master team in order to reduce the number of people reporting directly to him, freeing up his time to focus on the lead developers. This resulted in making a redundancy and hiring a delivery manager to manage the three remaining scrum masters. This was not a popular decision with the scrum master group, many of whom saw this as a demotion. Over the following year all of the scrum masters resigned, citing the return to command and control and a move away from agile working as significant reasons for their departure.

This period in SoftwareCorps’ journey is labelled ‘return of command and control’ as it depicts the period in which participants described a decay in agile working and a move back towards traditional command and control management. Communities of practice who had once worked together to make departmental decisions were disempowered and decision making was centralised under Julius (CTO). In addition to this SoftwareCorps’ acquisition by Tech Inc. created looming uncertainty around job security and business direction whilst simultaneously increasing pressure to deliver results and increase the number of projects in flight. This period also marks the end of the researcher’s time at SoftwareCorps and the observation of their journey.

In the following chapter the phase one findings of this research project are presented. This includes an analysis of the 12 agile principles and associated working practices as well the factors that influenced the adoption of agile working practices at SoftwareCorps.

# Phase One Findings – Agile Principles

This chapter presents the phase one findings of this study through an in-depth analysis of qualitative interview data and participant observations. In doing so, this chapter addresses *RQ1: How do the practices of agile software development enable the manifestation of agile principles in software development teams?*

Data collection for phase one of this study took place during the “agile by the book” (2013-2016) period illustrated in Chapter 5, Figure 7. This period commenced with the introduction of a new CTO role, first held by Jamie and subsequently by Nelson. This period saw the removal of existing project and development processes and the introduction of a ‘technology transformation’ aimed at establishing agile working across the organisation.

The chapter is structured as follows; First, a summary of the findings is introduced at a high level. Second, each of the 12 agile principles is analysed to provide a more in-depth examination of the working practices associated with each principle and the level of adoption of these practices by software development teams. Third, findings are presented regarding the factors that influenced the adoption of agile working practices. To conclude, the emergence of leadership as a significant influence on the adoption of agile at SoftwareCorps is described.

## Introduction

As has been detailed in Section 4.6, a set of a-priori codes were created before coding transcribed interviews. These codes were created based on the 12 agile principles in order to capture data around perception and manifestation of agile principles and practices in the day-to-day activities of teams at SoftwareCorps. The ordering of the agile principles in this chapter is driven by the data, as shown in Table 15 there were 27 participants in phase one of this study. The top 5 principles were self-organising teams, deliver working software frequently, reflect and improve, cross-functional teams, and satisfy the customer. These were mentioned by at least 70% of participants (20 or more) and were found to be the principles that participants recalled most commonly. For example, they were often recalled when defining agile and describing what it meant to work in an agile way.

Agile is often promoted as a way to achieve higher customer satisfaction and increase the frequency of delivering working software. It is unsurprising, therefore, to find the principles of satisfying the customer and deliver working software frequently rank highly. These two principles are somewhat self-explanatory, however self-organising teams, cross-functional teams and reflect and improve are more nuanced, and arguably less easily defined. Consequently, a brief description of these principles will be provided.

A **self-organising team** is autonomous and empowered. Members perform highly related work, are a social unit, are given authority and responsibility for their work and make decisions with economic consequences (Moe et al., 2008). In order to be autonomous, self-organising teams must also be **cross-functional.** A cross-functional team have all the competencies they need to accomplish their work within the team, so they do not depend on anyone outside of it (Schwaber & Sutherland, 2017). The principle of **reflect and improve** refers to a team inspecting their performance, identifying potential improvements, and creating a plan to implement them (Schwaber & Sutherland, 2017). Critically, the team must act on this reflection and continuously improve.

Table 15: Data-Driven Ranking of Agile Principles

|  |  |  |
| --- | --- | --- |
| **Rank** | **Agile Principles** | **No. Sources** |
| 1 | The best architectures, requirements, and designs emerge from **self-organising teams** | 27 |
| 2 | **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference for the shorter timescale | 25 |
| 3 | [**Reflect and Improve**] At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly | 24 |
| 4 | [**Cross-functional Teams]** Business people and developers must work together daily throughout the project | 22 |
| 5 | Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software | 20 |
| 6 | **Welcome changing requirements**, even late in development. Agile processes harness change for the customer’s competitive advantage | 18 |
| 7 | The most efficient and effective method of conveying information to and within a development teams is **face to face conversation** | 17 |
| 8 | Agile processes **promote sustainable development.** The sponsors, developers, and users should be able to maintain a constant pace indefinitely | 16 |
| 9 | Build projects around **motivated individuals.** Give them the environment and support they need, and trust them to get the job done | 15 |
| 10 | Continuous attention to **technical excellence and good design** enhances agility | 14 |
| 11 | **Simplicity** – the art of maximising the amount of work not done - **is essential** | 12 |
| 11 | **Working software** is the primary measure of progress | 12 |

To understand how, and to what extent, agile principles were adopted, focus was given to the working practices associated with them. As detailed in Chapter 2 (Section 2.3.2), a representative set of commonly used agile working practices was gathered from the literature. These practices were cross-referenced with data from the case under study to identify shared practices and to determine how they contributed towards the adoption of agile principles.

There are two terms used to describe the way working practices are adopted in this section; behavioural and cognitive. This categorisation aligns with similar labels found in the literature, such as Leroy and Ramanantsoa (1997), Kostova and Roth (2002), and Cram and Newell (2016). When a working practice is referred to as behavioural, this denotes a practice which has been carried out in action but does not form part of the individual's values or belief system. Conversely, if a working practice is described as being carried out cognitively, this is used to express actions which are being carried out because the individual's values and beliefs are aligned with the consequences of actions associated with these practices. For example, an individual who behaviourally attends a daily stand-up may be physically present, but not contribute in a way that allows the purpose of the daily stand-up to be fulfilled (i.e. for the team to optimise team collaboration and performance in order to maximise the probability of achieving their goal). Conversely, an individual who has transformed their beliefs and values to align with agile working practices would not simply go through the motions of said practices, but focus on realising their ultimate purpose having undergone a cognitive shift. These terms are used to elucidate an important distinction which was found to be evident in the way different teams and individuals carried out agile working practices at SoftwareCorps.

In this section, a representative set of practices is given under each principle to demonstrate how they were operationalised day-to-day. There is, however, overlap with many practices being associated with more than one principle. It should be noted that the practices given are not an exhaustive list of the practices used to achieve each principle in practice, but rather are those which give the most representative view of what was found at SoftwareCorps. In addition to this, complementarities between the principles are observed, such as between self-organising teams and reflect and improve.

Each of the 12 agile principles, the associated working practices, and level of adoption will now be explored in more detail. They will be discussed in the same order as presented in Table 15, as driven by the data. To demonstrate how leadership emerged from the phase one findings this will be explicitly described under each of the factors influential to agile adoption, although this will be covered in depth in Chapter 7.

## Self-Organising Teams

The principle of *'the best architectures, requirements, and designs emerge from self-organising teams'* ranked top of all the agile principles based on the data. Both direct references to and indirect examples of self-organisation were mapped to this code. All 27 participants from phase one spoke in some way about the concept of self-organisation in relation to their practice of agile, with 120 individual references coded.

### Practices associated with Self-Organising Teams

There were several working practices associated with self-organisation in teams at SoftwareCorps, which fell under behavioural or cognitive practices. Behaviourally, the daily stand-up was seen to provide an opportunity for the team to come together and self-organise around the day's work. Where teams at SoftwareCorps had achieved a level of self-organisation, the stand-up was a team owned event that focused on the team's challenges and goals for the day. Other practices which facilitated self-organisation included establishing some level of cross-functionality on teams, with embedded product owners, which reduced dependencies on external parties. This structure enabled the team to make day-to-day decisions without having to reach outside of the team.

Additionally, co-locating all team members in an agile workspace with breakout areas, whiteboards, and an area for the team to do stand-up around their scrum board, facilitated better collaboration and communication. These were seen to be essential for effective self-organisation. Examples of other behavioural practices included the use of burndown charts and velocity tracking as team measures, and providing data for the team to review their progress and self-correct where needed. In addition to providing the team with a mechanism for monitoring progress, these practices gave them the ability to predict when features were likely to be delivered. This prediction was given as a range rather than an absolute date. These measures enabled them to take responsibility for delivery and allowed them to own their deadlines, setting realistic expectations for the business.

Agile working practices that required individuals to make cognitive changes included transitioning to be multidisciplinary, collective code ownership and knowledge sharing. In order for teams to effectively self-organise, all members of the team needed to be willing and able to contribute towards any team problem or task. This required individuals to branch out from their given specialism and become multidisciplinary - increasing their knowledge of other disciplines. At SoftwareCorps, this was seen most frequently in developers who strived to improve their testing skills, and in testers who developed their coding skills to write automated testing frameworks. This upskilling allowed the team to organise around the work to be done to meet their shared goal, rather than defining the work to be done based on the available specialists on the team. More details and examples of this are given under Cross-Functional Teams on page 175.

The practices of collective code ownership and knowledge sharing were seen to go hand in hand and were facilitators to self-organisation as they created a level playing field for all team members. Collective code ownership, a term originally from Extreme Programming (XP), means that the entire team owns the code base and anyone can make changes to any area. This shared ownership created an environment where no area was out of bounds, addressing historical attitudes that seniority meant greater ownership of the code. Collective code ownership was complemented by the practice of knowledge sharing, as those with more experience shared their knowledge to ensure the success of the team as a whole. In some cases, this was done through pair programming, in other cases face-to-face code reviews or group design discussions. At SoftwareCorps, there was a focus on giving team members a broader knowledge of their products and clients, encouraging individuals to learn other disciplines while still appreciating the need for a level of specialisation. Having multidisciplinary individuals gave the team greater flexibility around who worked on each task and therefore created durability of the team when individuals were on holiday, sick or left the team.

Finally, self-organisation at SoftwareCorps was also seen through the freedom afforded to teams to define their working practices and experiment with different ways of working. This practice created an environment where the team felt they owned their processes and could adapt and change them based on their context and needs. The majority of teams utilised their end of sprint retrospectives to reflect on how they could improve their processes and suggest experiments for future innovations in how they worked.

### Adoption of Self-Organising Teams

In terms of the adoption of self-organisation in teams at SoftwareCorps, this was a subject frequently spoken about at the team and departmental level. However, opinion was divided amongst the participants regarding the implementation of self-organisation and its effects.

Positive reports of self-organisation addressed both the idea in principle and its manifestation. Teams who were free to self-organise were reported to take greater responsibility for delivery. Participants spoke positively of leaders empowering those doing the work to decide how it should be done, as well as the benefits of tackling problems as a team rather than individually. This reportedly led to better solutions. Participants also associated self-organisation with an environment where all voices on the team were heard and all ideas valued. They also reported having the opportunity to learn from their own mistakes which created a feeling of empowerment and responsibility.

"The team itself makes the decisions. I don't do it sort of top-down, dictatorial style. You know, we decide within the team, people have responsibility within the team to collectively come up with the right solutions. But, I think that I make sure it's going in the right direction and guide it" (Woody, LD)

On the other hand, some argued that senior individuals (seniority in terms of experience rather than formally titled) would naturally have more weight on a team, undermining their interpretation of self-organisation which considered all team members equal. Some felt that self-organisation was being mandated from above, which was seemingly contradictory. Furthermore, some saw self-organisation as a fashionable agile buzzword with little to do with developing software. As such, they were cynical it could be achieved in practice.

"I think self-management only goes to a certain degree. Complete self-management is an illusion because you could only achieve it if everybody in the team is very senior" (Julius, CTO)

Challenges to self-organisation were seen where there were lead developers who perceived their role as managerial. Consequently, their presence and actions suppressed self-organisation on teams with whom they worked. Challenges were also faced where teams attempted to influence aspects of their environment which sat outside of the technology department. While a level of self-organisation was evident within the boundaries of the department, many teams faced organisational constraints when attempting to instigate change, in particular with the delivery department with whom they needed to work closely.

As was evident at SoftwareCorps, self-organisation was not something a team could choose to do autonomously. It required buy-in and support from leadership as well as the wider organisation. With this in mind, although teams could be seen to enact the practices discussed in this section, this alone was no indication that they were a self-organising team. Although these practices were seen to facilitate self-organisation, they were not enough on their own. Without supporting structures such as flattening the hierarchy (removing titles such as 'Junior' and 'Senior'), allowing teams to make decisions (such as regarding how they worked) and moving away from 'managing people' to coaching, facilitating and empowering them, self-organisation could not be considered adopted.

## Deliver Working Software Frequently

The second most referenced agile principle based on the data was that of *‘deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale’*, with 25 of 27 participants referencing this in some way.

### Practices associated with Delivering Working Software Frequently

There were several practices associated with the principle of delivering working software frequently, including adopting two-week timeboxed 'sprints', with each sprint resulting in a 'potentially shippable' product increment. At SoftwareCorps this meant that, at the end of each sprint, the team aimed to have all stories 'done' by the team's definition of done[[5]](#footnote-5)\*, including all development, testing and documentation; in other words, there should be nothing left for the team to do in order for that piece of functionality to be shipped. It was then down to the product owner to determine whether there was value in shipping this product increment to customers, or whether to wait for further increments before releasing.

Other practices included changing the way requirements were defined, breaking them down in such a way as to enable functionality to be released independently, through the use of ‘user stories’, or ‘stories’ for short. Frequently used alongside Scrum, although originating from XP, a story is defined as the smallest possible deliverable which meets the following criteria; it is Independent, Negotiable, Valuable, Sized-appropriately and Testable (the acronym INVEST is used to denote these attributes).

Within teams who were striving to release as frequently as possible (as a minimum once per sprint) changes were also made to testing practices. Firstly, teams focused on test automation which shortened the feedback loop between development and testing. It was reported that by building reliable and fast-running automated tests covering the core functionality of a product, the team could monitor when breaking changes were introduced. This monitoring enabled them to respond more quickly to bugs and avoid release blockers towards the end of a sprint. Secondly, accountability for quality and testing was shifted away from being the role of individuals (such as testers) and framed as a responsibility for the entire team, necessary for them to achieve their shared goal. Shifting the focus for testing and quality away from a subset of individuals and giving shared responsibility to the team led to developers considering testability and test automation at the point of development, rather than as an afterthought. As a result, 'testing backlogs', caused by having significantly more developers than testers, were reduced. This ultimately allowed the delivery of working features sooner.

### Adoption of Delivering Working Software Frequently

In terms of adoption levels of the working practices discussed in this section, and by association the principle of delivering working software frequently, this varied between teams and across projects.

A majority of participants supported the principle of delivering working software frequently. In particular, product owners were keen to build incrementally and release early. This would allow them to validate assumptions and gather feedback as early as possible, enabling them to identify any quality problems quickly and respond to changing requirements. Several participants spoke of aspirations to be able to release at any point in the project, and that regularly releasing had been seen to increase confidence, not only in the product but also the quality of the development. It was also reported that this regular delivery provided a higher level of satisfaction in the software development team who saw their features 'get out of the door', creating a greater desire in developers for regular releases.

"If you want to build a business, you've got to deliver working software all the time…you keep everybody focused on delivering value...You've got to release. You can theorise all you want, but get it into your client's environment and keep it working and keep that traction with your clients" (Paul, SM)

However, not all feedback was positive. One concern surrounded the need to break larger pieces of work down into smaller, independently releasable slices in order to achieve the regular delivery of working software. Some participants argued that the perceived additional overhead involved in breaking work down in this way was unnecessary for some projects and could be cumbersome. Projects where the customer would not use the product until all features were complete were given as examples of this. Also, the re-write of an existing product where the requirements were unchanging, and there were no functional differences between the old and new product.

Participants also spoke of a lack of buy-in across the organisation, such as in the customer-facing departments who reportedly did not see the value in the ‘little things’, preferring to deliver an entire product rather than a series of smaller features;

“[The business are] not seeing the value in little things going out…when it comes to [incremental releases] they’re like “Yeah, but this isn’t really going to be useful until we have blah. This isn’t really going to be useful until we have blah.” (Emmett, LD)

This lack of buy-in was re-enforced by a lack of appetite from some customers for regular releases due to their substantial change control processes which hindered the acceptance of regular software updates.

Teams within the software development department were split into four 'quadrants' focusing on specific products. Between the four quadrants, all were seen to adopt the practice of working in two-week sprints. However, only three were seen to ship working software regularly. There were periods, however, where external factors blocked even teams who had achieved regular delivery of working software. For example, one team had a series of releases which were found to have bugs in production, and consequently, the delivery team increased the level of governance required for each release, reducing the frequency at which the team could ship products.

The fourth quadrant, who were responsible for SoftwareCorps oldest products, did not regularly ship working software despite working in 2-week sprints. They described several blocking reasons for this, including a lack of appetite from their product owner to define independently releasable requirements, which was reportedly a consequence of their clients having lengthy change control processes which were incompatible with accepting fortnightly releases. Additionally, they reported that the complexities of working with a 'legacy' C++ codebase made test automation and delivering working software every sprint almost impossible due to the exiting architecture and testing overhead required for each change. They described this legacy codebase as having many "dark corners" and "cobwebs".

Consequently, although a majority of teams at SoftwareCorps adopted the practices associated with delivering working software frequently, it could not be said that this was fully embraced by all teams or on all projects. While some teams adopted the practices associated with this principle, this did not always lead to them achieving regular delivery of working software.

## Reflect and Improve

Ranked based on the data as 3rd of the 12 agile principles, *'at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly'* was mentioned in some way by 24 of 27 participants with 106 individual references. The majority of participants associated this principle with their team retrospectives which were carried out at the end of each 2-week sprint. As such, references to retrospectives were mapped to this principle during coding.

### Practices associated with Reflect and Improve

Much of the way teams at SoftwareCorps worked (through following Scrum) encouraged active and regular reflection and continuous improvement. For example, a key ceremony in Scrum is the team retrospective. Its purpose is to facilitate continuous improvement of the team's processes. Teams would use other practices such as burndown and velocity, the sprint demo and Kanban to gather data and visualise their process in order to identify areas for improvement. Teams who had a good idea of their velocity historically were able to use this to assess the impact of changes to their working practices, environment and team configuration. A number of teams were seen to review their burndown chart, using it as a tool to prompt conversation on how the team might improve going forward, or to reflect on how they had handled disruptions to their process. The sprint demo, which was a company-wide event, was also used to facilitate team improvements through gathering feedback from outside the team. Of particular importance was the emphasis on experimentation and teams were encouraged to try different ways of working in order to innovate and improve.

### Adoption of Reflect and Improve

Considering the level of adoption of these practices, and in turn the principle of reflect and improve, it is necessary to distinguish between the ceremony of the team retrospective (behavioural) and the act of reflection as a driver for improvement (cognitive).

There were varying reports of how valued and effective retrospectives were from team to team. Some participants argued that the time taken to carry out a retrospective was seen by some product owners as costly, suggesting that they would prefer the time to be used for development of the product. Other participants questioned the effectiveness of retrospectives that did happen. There were reports that some teams used their retrospectives to air grievances, but that the same issues would be brought up repeatedly, with little to no action taken to address systemic problems. Where teams were unable to drive change, for example when problems were too big for the team to tackle alone, this led to disengagement from the practice of retrospectives;

“It was just a case of you were repeating yourself because the things that were an issue for you two weeks ago were still an issue for you. So we hadn't done anything with them, and nobody really wants to sit around and talk about the same problem" (Adrian, SM)

A majority of participants, however, spoke positively about the principle of reflect and improve, and about the act of carrying out retrospectives. Participants saw value in the practice of regularly reflecting on team processes and how the product was developing. Some participants suggested that even if teams did not make any changes, it was a good opportunity for individuals to voice their grievances and move past them. Participants also associated effective retrospectives with improving their productivity, using the fortnightly cadence of the retrospective to experiment with new ways of working in order to improve and learn. They suggested that regularly reflecting and improving helped teams to embrace change, take a measured approach, and adopt a mindset of continuous improvement where mistakes were seen as an opportunity for learning.

“People make mistakes because…perhaps that’s the only way they can see of solving the problem when there’s other ways of doing it and sometimes it’s good for people to actually burn themselves” (Emmett, LD)

As has been a common theme throughout this chapter, a distinction was identified between following a process behaviourally, such as going through the motions of team retrospectives, and effectively using agile working practices to embrace agile principles – making a cognitive shift. Although all teams at SoftwareCorps were seen to carry out retrospectives as part of following Scrum, not all teams could be said to have made a cognitive shift to adopt the principle of reflecting regularly and tuning their behaviours accordingly. To embrace the principle of reflect and improve, rather than just enact the ceremony of a retrospective, a cognitive shift is required from individuals.

## Cross-Functional Teams

*'Business people and developers must work together daily throughout the project'* summarised for brevity here as 'cross-functional teams' ranked 4th of the 12 agile principles. It was mentioned by 22 of 27 participants, with 81 individual references. The definition of cross-functional was understood by participants to be a team who had all the skills required to deliver a feature end-to-end, from ideation through to delivery. In the context of SoftwareCorps, this meant a team would need a member of product, technology, sales and delivery to be embedded within a single team, working together towards a common goal.

### Practices associated with Cross-Functional Teams

Practices which contributed towards this principle included encouraging individuals to be multidisciplinary, creating knowledge sharing teams, utilising shared goals, dedicating a product owner to each team, and co-locating team members.

To reduce knowledge silos, individuals at SoftwareCorps were expected to be somewhat multidisciplinary. That is, although each person had a specialism, they could carry out tasks in other disciplines. For example, developers were expected to be able to participate in testing, and testers were expected to learn enough about coding to write automated tests. Upskilling, in this way, helped the team to balance their workload and also reduced knowledge silos.

Problems with knowledge siloes were seen to occur in teams at SoftwareCorps where certain types of work or certain tasks could only be done by a specific team member, who knew that area. If that team member was out of the office or tied up on another task, this created queues of work. These queues reduced the team's ability to deliver as it created dependencies. An example of this was seen in what was referred to as a 'testing backlog', where the amount of testing to be done by the relatively small number of testers was growing faster than it could be completed. For example, one team at SoftwareCorps were aiming to release a specific feature at the end of their sprint. To do this required all team members to participate in testing as there was a considerable amount to be done and a limited number of testers. The team rallied around the work to be done, rather than having those who specialised in development moving on to developing the next lower priority feature. All team members agreed that it was preferable for the team to swarm around the highest priority item and get it out, rather than have several partially complete items in flight concurrently. By facilitating and encouraging team members to branch out into other disciplines, teams were seen to become more durable and were able to flex based on the priorities of incoming work.

"What we've seen is that people need a broader skill set for agile, so we try to work in a way where anyone on the team can do everything almost, so we don’t have very clear roles between developer and tester. There are people that focus on testing and people that focus on development because that’s what their strengths are, but the team should be able to help out in the other role when it’s needed” (David, PO)

Where teams adopted a multidisciplinary approach at SoftwareCorps, there was a notable shift towards finishing rather than starting stories, with team members seen to ask at stand-up what they could do to help complete in-progress stories before picking up a new one.

Having multidisciplinary team members was an essential practice of cross-functional teams at SoftwareCorps. Through this practice, individuals from across various disciplines (business and technical) worked closely together, increased knowledge sharing across the team, and created an environment where everyone 'mucked-in' to get the job done. Ultimately this benefitted teams by bringing business and technical people together, increasing collaboration and empathy for different roles and ensuring all parties were working towards the same goal.

“So with bigger organisations and more distinct roles people sometimes have a silo mindset; so they know their role, their world and what they need to get done, but maybe don’t have as much understanding of what the world around them needs. So from that perspective [cross-functionality is] very helpful…it makes sense to have roles that are maybe not very strictly defined, but somewhat defined and make sure that people work together to the same goal” (David, PO)

At SoftwareCorps each team had a dedicated product owner who was responsible for ensuring the needs of the business and customer were understood by the rest of the team, which allowed them to make better day-to-day technical decisions within the context of these needs. The product owner was the embodiment of ‘business people and developers working together daily throughout the project’.

The final practice to be discussed concerning this principle is that of co-location. All members of the software development team (developers, testers, product owners) at SoftwareCorps were seated together, with access to breakout spaces and whiteboards for informal and impromptu team discussions. This co-location was especially important on cross-functional teams as it facilitated ad-hoc collaboration between individuals who traditionally did not interact. Communication and collaboration were seen to happen organically between co-located individuals working towards the same goal, as team members overheard conversations they might otherwise not be privy to. In a similar vein, co-location facilitated knowledge sharing, as individuals were in close proximity and therefore could more easily ask questions or overhear conversations which contributed towards them gaining a greater understanding of the product, customer and codebase.

### Adoption of Cross-Functional Teams

In terms of the adoption of cross-functional teams at SoftwareCorps, all software development teams were cross-functional to some extent, with developers, testers, scrum masters and product owners on every team. However, cross-functionality outside of this was only seen to exist for short periods, with the secondment of individuals from other departments for specific projects or tasks. For example, during a period of regulatory change which required modifications to an existing product, a short-lived team with members of product, technology, sales and delivery was created who were co-located and worked together daily until the project was delivered.

Opinions varied regarding the need and effectiveness of these 'fully' cross-functional teams. There were concerns raised by those who had been members of short-lived cross-functional teams, usually created for a specific project and subsequently disbanded, that prioritisation was less objective. A lack of objectivity was reportedly due to members of sales and delivery being responsible for specific clients, which created a tendency for them to push for prioritisation of 'their' client's features over the needs of the broader customer base. Another issue reported from these short-lived teams was that members of different functions would vie for control, with power struggles between individuals who believed they should govern the team;

"it was then no longer about 'if we have cross-functional teams we are much more efficient' than it was 'yeah let's have cross-functional teams, but of course my department dominates them or rules them'" (Julius, CTO)

In addition to reported power-struggles within the team, there were also concerns that motivations for creating fully cross-functional teams were driven by broader political issues, with leaders seeking to 'land grab' and increase their sphere of influence. There were also economic concerns regarding the sustainability of teams with members from each function. Some feared that this created overly large teams with individuals who were not required full-time to carry out their specific role. Although this concern could be addressed if individuals were multi-disciplined and able to fulfil other roles.

Despite these difficulties, from the limited practice of creating cross-functional teams at SoftwareCorps, many participants described them as more effective than their siloed counterparts. Cross-functionality reportedly created greater alignment between functions where members of different departments worked together as a single team, as well as reducing conflict between them as their focus was on a common goal. Cross-functional teams reportedly achieved faster delivery as the team was able to control the delivery process end-to-end rather than relying on an external team to pick up the baton. This increased ownership addressed historical 'throwing it over the wall' mentalities which referred to the team shirking responsibility before a feature was fully delivered. Other positives reported from short-lived cross-functional teams were that individuals achieved an appreciation for other roles and reduced knowledge siloes, which built trust and created stronger relationships between functions.

"For a while [a delivery person] was even sitting with the development team and he knew all the issues that came up during development, he got a sense of when the release was getting out, so in his mind, he could already prepare that this is going out…he feels like part of the team, and he just wants the entire, wider team to be successful and that's very beneficial compared to having a wall between delivery and technology where technology finish something, they write a release email and then delivery don't really have a sense of what's in that release and why they should ship it to anyone" (David, PO)

Although all software development teams at SoftwareCorps included a product owner, scrum master, developers and testers, few teams were fully cross-functional in terms of having all the skills required on the team to deliver a feature end-to-end. Only in instances where there was an acute need for fast turnarounds, such as a regulatory change, were 'fully' cross-functional teams created and as such it could not be said that the principle of cross-functional teams was fully adopted at SoftwareCorps.

## Satisfy the Customer

Ranked 5th of the 12 agile principles*, 'our highest priority is to satisfy the customer through early and continuous delivery of valuable software'* was mentioned by 20 of 27 participants, with 76 individual references.

### Practices associated with Satisfy the Customer

There were a variety of practices associated with the principle of satisfying the customer, many of which focused on making it possible for teams to quickly and iteratively deliver software in order to engage the customer and gather feedback. For example, feedback loops (the time between building a feature and the team receiving feedback regarding its use) were shortened by releasing more regularly to customers. Opportunities to adapt based on incoming requirements were also increased by working in two-week sprints (a new requirement was never more than two weeks away from being picked up). Additionally, the product owner was the physical representation of the customer to the team, ensuring that the customer's needs were at the forefront of day-to-day development.

"If you don't do releases often it looks like we've gone dark and we're not doing anything, and you want to say "No, we care about you. We're doing this. Here it is. Is this what you want? Do you want this?" and then you get lots of feedback." (Emmett, LD)

Defining requirements in the form of 'user stories' also ensured teams discussed technical aspects of the requirement in the context of user needs, focusing on the purpose of a feature and why it is of value to a customer. Stories were, by definition, small, valuable, and feasibly do-able within a sprint. Working with stories in this way meant that short iterations delivered value through frequent, small releases.

In order to shorten the feedback loop between building a feature and testing it fulfilled the acceptance criteria, teams invested in automated acceptance testing and test-driven development (where the developer writes a test first before writing the code to make the test pass). This practice helped on two levels. Firstly, it ensured developers considered the customer's needs throughout development, and secondly, it helped reduce the feedback loop between development and test, reducing the number of bugs found late in development which in turn helped increase the frequency at which the team released. Writing automated tests to accompany every story helped identify bugs earlier in development, as well as safeguard features against potentially breaking changes in the future. Where teams did not invest in test automation, a far greater burden and reliance was seen on manual testing at the end of development, which was often seen to delay releases when bugs were found late in the sprint.

All product owners at SoftwareCorps maintained a product backlog, which consisted of user stories at varying stages of definition. Stories at the top were well-defined, whereas those at the bottom were less well-defined. The backlog was kept visible to the team, department, and wider business through the use of a common tool – Team Foundation Server. Through this tool, teams were able to manage their sprint backlog day-to-day, and have visibility of the wider product backlog and upcoming stories. This practice was accompanied by working towards a rolling 3-month company roadmap which was displayed on monitors around the office, as well as a physical roadmap-level Kanban board positioned centrally in the office. All of this was used as a way to ensure visibility of the current and future roadmap, contextualising the day-to-day work teams did in relation to the bigger picture of satisfying wider customer needs.

### Adoption of Satisfy the Customer

In terms of the level of adoption of these practices, all teams utilised user stories, worked in two-week sprints and made their product backlogs visible through the use of TFS. However, as was described under Deliver Working Software Frequently on page 169, not all teams delivered working software each sprint. As such, these teams could not be described as delivering 'early and continuous value' to satisfy customer needs.

For example, the application of test-driven development (TDD) was not mandated, and therefore became a personal choice for each developer based on their experience and preference. It was found that only a small number of individuals honestly wrote tests first, with many adopting a looser interpretation and writing the tests after the body of development was completed. There were also variances between teams in terms of the coverage of automated acceptance tests, with a tension reported between those who believed test automation to be essential and those who believed it was too costly. For example, some product owners were seen to de-prioritise writing of automated tests when deadlines were looming, or to pressurise developers into limiting writing tests where this was part of the team’s definition of done.

When speaking of agile working, it was common for participants to describe agile as having a focus on regularly delivering value to customers and meeting customer needs. There were, however, several challenges reported, including the nature of the products SoftwareCorps offered and the overarching company strategy. A variety of customers used SoftwareCorps' products, and as such, it was necessary to consider the overall requirements of the market, rather than the individual needs of each customer. This perspective meant that features which benefitted the most customers would be prioritised over those that served only a smaller number, which could lead to some customers' requirements not being met. Conversely, there were also cases where features were prioritised for individual customers in order to broker a larger deal, at the cost of meeting the needs of the wider customer base, echoing statements from some participants that customer satisfaction came second to business profits;

“[SoftwareCorps’] whole business model is milk the cash cow, right? It’s take this piece of kit that we’ve got and make as much money as we can from it and spend as little on developing as possible” (Harper, PO)

In summary, not all teams adopted the practices required to increase customer satisfaction, such as shipping value to customers every sprint, and it was observed that the wider company strategy could ultimately conflict with this principle.

## Welcome Changing Requirements

Ranked as 6th of the 12 agile principles, *'welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage'* was mentioned by 18 of 27 participants, with 41 individual references.

### Practices associated with Welcome Changing Requirements

Working practices which enabled teams to respond to changing requirements included working in two-week sprints and demoing the product at the end of each sprint, ensuring the product *“never goes more than two weeks in the wrong direction” (Woody, LD).* Some teams were also reported to focus on keeping as close to releasable as possible throughout their sprint, enabling them to ship what they had worked on should priorities change, minimising waste. The use of user stories, which are described as 'placeholders for a conversation' in Scrum, also encouraged the welcoming of changing requirements as they replaced traditional business requirements specifications and defined the customer's problems, rather than defining solutions. Focusing on requirements allowed the solution to change as more was understood about the customer's needs.

It was a commonly held view within teams at SoftwareCorps that requirements would always change over time. However, where it was known that requirements were likely to be particularly volatile, some teams chose to follow Kanban rather than Scrum. Kanban allowed the product owner to change the ordering and priorities of the requirements in the backlog up to the point a story was picked up (compared to Scrum where the team select a set of stories to form their sprint backlog which should not change after the sprint has started). It was seen as best practice that once started, a story should be completed rather than switching between stories and having to 'shelve' changes if priorities changed, however as most stories were small by definition, this did not pose a significant problem.

### Adoption of Welcome Changing Requirements

As has been noted in previous sections in this chapter, all teams worked in 2-week sprints, utilised user stories to define requirements, and held company-wide end of sprint demos. As described in this section, these practices shaped the processes teams used to equip themselves to handle change and helped to create an environment where uncertainty was expected. The practice of staying close to releasable at all times was adopted less consistently. Participants spoke of this in an aspirational way rather than describing it as their current approach. As was described under Deliver Working Software Frequently on page 169, there were several challenges to being able to release 'at any point'. The complexities and technical debt of the incumbent codebase, the level of existing test automation, the likelihood that the team would need to release at any point, and whether the customer would accept regular releases were all factors.

The willingness of teams to accept changing requirements was seen to vary depending on individuals and historical context. While most participants agreed that agile working practices were designed around the idea that requirements are likely to change over time, there were also concerns that this agile principle was often misused to allow for more systemic organisational failings. It was argued that although agile working is predicated on responding to change, this could be used as an excuse in some cases for inadequate requirements gathering and product planning;

“I think it’s a misinterpretation...with agile we mean we have a process that’s built on the assumption that requirements change over time and what...the rest of the company took away [is that] we are now agile we can change what we want do on a daily basis, and this is not agile this is just a waste of energy and time” (Julius, CTO)

Where individuals had been subjected to this ‘misuse’ of the principle, they were seen to be less welcoming of requirements changes and more likely to object or suggest excessive requirements definition in an attempt to force better product planning.

It was also suggested that some teams found welcoming changing requirements costly and disruptive. This was especially true when new features required significant re-writes of existing code, although these drawbacks were seen as palatable by most in the context of the broader benefits. These benefits included increased responsiveness and the ability to change dynamically as more was understood about the customer's needs.

In summary, as all teams adopted some form of Scrum or Kanban. Given these methods assume that requirements will change over time, it could be said that teams were equipped to accept changing requirements. However, not all teams could be said to welcome changing requirements, with some individuals arguing that changes were often a consequence of poor product planning.

## Face to Face Communication

Ranked as 7th of the 12 agile principles, *'the most efficient and effective method of conveying information to and within a development teams is face to face conversation'* was mentioned by 17 of 27 participants, with 33 individual references.

### Practices associated with Face to Face Communication

There were several practices which encouraged face to face communication to and within the software development team, such as the daily stand-up where each team gathered physically around their scrum board to discuss the days' work. Individuals who were not physically present that day were dialled in on video conference to enable them to participate in an *almost* face to face manner. Other team ceremonies such as the end of sprint retrospective, demo and sprint planning were carried out face to face, with any remote team members travelling into the office on sprint changeover day when most of these activities took place. Additionally, importance was placed on co-locating teams. Particular thought was given to the arrangement of seating to facilitate easier communication between team members. As such, breakout spaces were created where informal design discussions could be held around a whiteboard. Finally, client visits were established for developers, enabling face to face interactions with real users, resulting in a greater appreciation of customer needs.

“I am introducing customer visits. So that people get in touch with real [users], using their products in a real-world scenario, so they see how the software is used, they understand it, they can ask questions” (Julius, CTO)

### Adoption of Face to Face Communication

When it came to the adoption of face to face communication, the participants universally supported face to face interactions as the most effective form of conversing, not only within software development teams but across the organisation. It was reported that a focus on conversing face to face facilitated walls being broken down between departments, where previously there had been poor communication and strained relationships. Of particular importance was the shift away from providing requirements in the form of lengthy specification documents and towards face to face discussions between product owners and developers.

“The wall being broken down between [product owners] and developers is a good thing. Would I want to go back to having a big document laid in front of me and say ‘develop this’? No, not at all” (Darius, Dev)

The only downside reported as a consequence of increased face to face communication was that it could lead to difficulty focusing on a task if team members were frequently approached and asked questions. An email or electronic message, on the other hand, could be muted or ignored until the task was complete.

In summary, where teams were co-located, face to face communication was universally adopted as the preferred form of conveying information. Teams which included remote members used video-conferencing to bridge the physical gap, and remote workers travelled into the office once per sprint for team retrospectives, demo, and planning.

## Promote Sustainable Development

Ranked 8th of the 12 agile principles, *‘agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely’* was mentioned in some way by 16 of 27 participants, with 23 individual references.

### Practices associated with Promote Sustainable Development

There were a number of working practices associated with the principle of promoting sustainable development, many of which were taken from Scrum. At the team level, estimation, velocity, and sprint planning were used together to set realistic expectations regarding delivery. This was done through teams maintaining a backlog of prioritised and estimated future work, in the form of user stories. The further down the backlog a piece of work was, the less detailed the requirement and therefore the team's estimate was understood to be less accurate. By capturing data from their previous sprints, teams were able to gauge their velocity (how many stories they typically completed each sprint). Combining the estimated and prioritised backlog with the team's velocity the product owner, in conjunction with the team, could predict a range within which features were likely to be delivered, and communicate this at the organisational level. In this way, teams were seen to use estimation, velocity and the product backlog to establish a sustainable pace and set realistic expectations regarding delivery.

### Adoption of Promote Sustainable Development

At different points in time, the focus shifted towards or away from promoting sustainable development. Participants recalled a time when the business appeared to agree that it was unsustainable to fix the deadline, scope, quality, and resources on a project – and still expect a team to deliver. During this time (during the ‘agile by the book’ 2013-2016 period illustrated in Chapter 5, Figure 7), excessive overtime was discouraged and seen as a false economy leading to lower quality due to burnout. In order to shift away from externally set fixed deadlines, teams were encouraged to participate in product planning in order to help shape deadlines and scope for projects they worked on. This participation was seen to help move the conversation away from deadlines and towards effective prioritisation. Effective prioritisation meant that the teams were always working on the most important thing, delivering value every two weeks. Regular delivery of value reduced the need for 'big bang' project delivery dates.

However, changes in management and leadership (commencing the ‘undoing of agile’ 2017- period illustrated in Chapter 5, Figure 7) led to a period where there was an explicit expectation that deadlines (usually set externally to the team) were immutable. As such, participants reported feeling a high degree of pressure to deliver, often leading to working longer hours and feeling compelled to cut corners, consequently reducing quality.

“Well, it’s promises being made that we just cannot meet. For example, a thing we’re working on at the moment...was promised to the [customers] to be released before they’d even started work on it and before the scope was even defined” (Emmett, LD)

In summary, while there was a period in time (during the ‘agile by the book’ 2013-2016 period) where SoftwareCorps focused on promoting sustainable development, this principle did not endure over time, and although practices associated with this principle were carried out, they were not enough to ensure continued adoption of the principle.

## Motivated Individuals

Ranked as 9th of the 12 agile principles, *'build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done'* was mentioned by 15 of 27 participants, with 61 individual references.

### Practices associated with Motivated Individuals

As was described earlier in this chapter, self-organisation and cross-functional teams were both reported to increase the motivation of individuals to deliver and take ownership of their products. As such, practices associated with these principles (e.g. daily stand-up, co-location, knowledge sharing, retrospectives, sprint demo) were also found to apply here.

### Adoption of Motivated Individuals

Some participants suggested that this principle, in particular, was *“a bit fluffy”* (Reece, LD) and as such many had difficulty recalling the principle other than citing a focus on people as being important in agile working. It was evident that there was some cross-over between this principle and the principles of cross-functional and self-organising teams, in particular when considering a call to provide teams with ‘support’ and to ‘trust them to get the job done’.

Many participants acknowledged that it was paramount that individuals felt motivated. However, of particular note was the importance of understanding *what* motivated people, and accepting that this would vary from person to person.

"[People] come to work on things for a reason and, it doesn't matter what industry you're in, there is something that interests them, excites them and maybe it's purely the paycheque, but that's their driver, and that's personal to them and...it’s about trying to relate to their drivers even if you don’t understand them or agree with them” (Ted, LD)

When speaking of his role as a lead developer, Emmett explained that in order to motivate his team, he focused on ensuring the team had ownership of the problems they were trying to solve. Drawing on the participants’ accounts of working in self-organising teams, where trust and empowerment were key, it was evident that individuals on these teams took greater responsibility for delivery and were trusted to deliver, suggesting a higher degree of motivation.

“you really want them to feel like they own the problem, like their opinion is important and their ideas are valuable and that they can solve it and that they get satisfaction out of doing it.” (Emmett, LD)

In summary, although there were no additional practices associated with this principle (those which apply have been described in earlier sections), many participants associated agile working with increasing motivation. This motivation was achieved through a focus on creating autonomous self-organising teams who were empowered to own delivery of their products end-to-end. Although self-organising and cross-functional teams were not fully adopted across all teams and projects, partial adoption was seen and evidence of increased motivation reported.

## Technical Excellence and Good Design

Ranked as 10th of the 12 agile principles, *'continuous attention to technical excellence and good design enhances agility'* was mentioned in some way by 14 of 27 participants, with 32 individual references.

### Practices associated with Technical Excellence and Good Design

There were several practices associated with this principle, many of which originate from Extreme Programming (XP), including; code reviews, short build time, pair programming, continuous integration and continuous deployment, refactoring, iterative and automated testing, test-driven development, collective code ownership, and establishing clear coding standards. Examples of these practices at SoftwareCorps have been described throughout this section, and as such will not be repeated here, instead, an explanation of how these practices help teams achieve technical excellence will be given. It is clear from the fact that these practices are associated with other agile principles that technical excellence enhances agility, as stated in the wording of this principle.

Code reviews and pair programming require that someone, other than the developer who wrote the code, has seen, understood and sense-checked the code before it is committed to the codebase. This step encourages developers to talk through their solutions and keep them simple, which often leads to improvements overall. In particular, pair programming requires that both parties agree with a design, and as such, design becomes more of a focus from the outset.

Short build time, continuous integration and deployment, and iterative and automated testing all contribute towards reducing the feedback time between writing and running the code. The easier it is for developers to commit and verify their changes, the more frequently and iteratively these activities can be completed. If the task of building, integrating and running the automated tests is lengthy, this is obstructive to an iterative approach. To maintain technical excellence, developers must know as early as possible if a change has caused a problem. The smaller the change, the easier it is to track down and fix. Therefore, if small changes can be built, integrated, tested, and deployed easily and quickly, this enhances technical excellence.

The practice of regular refactoring encourages the team to keep the code clean and simple and facilitates continuous attention to maintainability. As the system develops, the team ensures that existing parts of the system do not deteriorate and technical excellence is maintained.

Test-driven development (TDD) is the practice of writing a failing test for a given requirement before writing any code. The minimum amount of code required to make the test pass is then written. Once the test is passing, the code is refactored to meet agreed standards. TDD ensures good design is considered upfront, with satisfying the user's requirements in mind. Writing the test first also helps to verify the code iteratively, which contributes towards technical excellence as bugs are identified earlier.

Collective code ownership, meaning there are no parts of the code off-limits to other members of the team, ensures developers can change any part of the code necessary for a feature. Meaning the design, implementation, and verification of the code can be owned by the individual making the change. Individuals are therefore responsible for the design and execution of that change which encourages close attention to technical excellence and design.

### Adoption of Technical Excellence and Good Design

In terms of the adoption of these practices, some teams explicitly followed XP, whereas others chose practices more selectively. For example, lead developer Woody set the example for his team by following XP closely and educating others on the team about the benefits of XP practices. He led by example when it came to pairing, TDD and collective code ownership and coached his team in how to use these practices to the team's advantage. However, lead developers on some other teams did not lead by example and did not themselves adhere to XP practices. In these cases, practices were less likely to be adopted at the team level, becoming more of a personal choice for each individual. As a minimum, all teams carried out code reviews, where an independent developer reviewed code changes before they were integrated into the main codebase. All teams also created team coding standards against which code reviews were carried out (including coding style as well as content).

Several participants contended that, in the past, a willingness from the business to invest in technical excellence and good design was lacking. Instead, there had been a tendency to focus on problems that could be solved expediently. It was felt that there was a lack of appetite to spend time on anything which was not delivering features for customers. Teams felt pressured into choosing the quickest solution rather than the most sustainable long-term solution, leading to an increase in technical debt. Technical debt refers to the cost of rework required as a result of choosing an 'easy' solution now, rather than a better solution that will take longer to deliver.

“There are several massive problem areas in the code. We know there are bugs hidden in there, we know that it’s a maintainability nightmare and they come up in retrospective all the time, but no one wants to fix them because it’s a lot of work and…we won’t get given the time” (Darius, Dev)

However, a shift reportedly occurred with the sponsorship of a 'stabilisation' project focusing on addressing technical deficiencies. One focus of this stabilisation project was the implementation of product monitoring to enable SoftwareCorps to be more responsive to issues in production. Monitoring subsequently became a requirement for all future products. Participants felt that a focus on product maintainability increased the speed at which teams could develop new features, minimising the risk of bugs in production and increasing the reliability of the software.

In summary, although there were some practices associated with this principle which were carried out by all teams (coding standards and code reviews), the adoption of other practices associated with XP and technical excellence varied based on how invested members of the team were in following them. In particular, the lead developer was influential in setting an example for the rest of the team, and the wider organisation in valuing technical excellence (in the form of product maintainability) as equal to adding new features.

## Simplicity is Essential

Ranked based on the data as joint 11th of the 12 agile principles (with *‘working software is the primary measure of progress’* which is discussed next) *‘simplicity – the art of maximising the amount of work not done - is essential’* was mentioned by 12 of 27 participants, with 27 individual references. As suggested by its low ranking, this principle was not frequently mentioned by participants. Some indirect comments were mapped to this principle, however, such as those by product owners who spoke of a desire to have individuals with a pragmatic mindset on their teams to help coach less experienced team members to find the simplest solutions to problems;

"It's good to have a mixture of people so that you can talk about what an ideal solution would look like, but in most cases, there's really no need and no business justification to go for that cathedral. Sometimes a simple house is good enough. And I think it's really key that the development team leads, the most senior people on the team, have a pragmatic mindset and they can steer people in the right direction." (David, PO)

### Practices associated with Simplicity is Essential

Practices associated with this principle included the use of 'agile documentation', meaning minimal documentation which was owned and maintained by the team based on their needs. User stories were a form of agile documentation used by all teams to capture the essence of customer requirements and formalise team discussions, replacing complex business requirements specifications. Additionally, the practice of backlog refinement was observed as an opportunity for the team to discuss requirements with the product owner and debate the most suitable solution as a team, with a focus on "keeping it simple" and agreeing to the minimal viable product (MVP). Additionally, the construct of working in two-week sprints, at the end of which the team were expected to deliver working software, encouraged finding the simplest way to satisfy the customer's requirements.

### Adoption of Simplicity is Essential

Regarding the adoption of this principle, it was suggested that in the past, there had been a tendency for developers to opt for more complex solutions as a form of personal development. Developers would reportedly 'show off' with complicated solutions, or pick new technologies for experimentation rather than purpose. When working in an agile environment, there was reportedly less opportunity for this to happen as teams discussed requirements, solutions and designs as a group and were guided by the product owner who ensured the business requirements were well understood. The nature of how these requirements were broken down into small chunks also encouraged a focus on simplicity, as larger, more complex requirements would be refined into more easily understood slices.

There were also comments made by Julius (CTO) that, historically, developers had been measured and rewarded based on development proficiency. This measurement encouraged them to prioritise growing their knowledge of technologies, tools, and frameworks, and to choose more complex solutions as a way of demonstrating proficiency. Consequently, Julius (CTO) emphasised the importance of setting clear objectives aligned with desired behaviours, not only competency.

In summary, this principle was adopted in some way by all teams. This was evident through the use of user stories, backlog refinement and frequent small releases which encouraged simplicity and a focus on finding the MVP. However, a focus on technical competency in terms of personal development was seen to encourage some individuals to opt for complexity.

## Working Software is Primary Measure of Progress

Finally, ranked based on the data as joint 11th of the 12 agile principles, *‘working software is the primary measure of progress’* was mentioned in some way by 12 of 27 participants, with 22 individual references.

### Practices associated with Working Software is Primary Measure of Progress

Practices associated with this principle included using velocity as a primary team measure (where only delivered stories were included), creating small and frequent releases at the end of each sprint and only demoing completed stories at the end of sprint demo. In order to deliver working software early in the project, teams took larger, complex requirements and would break these down into smaller, independently deliverable chunks of functionality, adopting a mentality of *“how do we get something working in the smallest possible increment?”* (Rowan, SM). When successful, this meant that teams working in sprints would deliver working software every two weeks and as such were able to demonstrate tangible progress to the wider business.

### Adoption of Practices associated with Working Software is Primary Measure of Progress

A common frustration on software development teams was a tendency for the business to promote, or even demand, that teams work on multiple products or projects at once. Outwardly this allowed them to report that 'progress' was being made on many fronts, with numerous projects in flight. However, this resulted in each project taking longer to deliver overall. Several participants highlighted that it was best practice in agile to limit the number of items in development concurrently and to allow teams to focus on delivering fully-functioning products end to end.

“it’s better to focus on one thing and do that well and then address the next thing and do that. So I think if we had a more old school approach or…way of thinking, then maybe we would have tried to do all of that in one go” (Joseph, PO)

As a result, there was a concerted effort from the software development department to limit the number of projects in flight concurrently, and on a smaller scale to encourage teams to ‘swarm’ on their sprint backlog in order to prioritise finishing stories rather than starting them. Teams described measuring success based on their ability to release to customers each sprint, and on meeting their sprint goal.

In summary, it was observed that within the software development department teams were measured, and measured themselves, based on their ability to deliver working software frequently and as such that this principle was adopted across teams in SoftwareCorps.

## Context

All organisations exist within an environment or context which ultimately influences them. This study is focused predominantly on software development teams. However, consideration must also be given to the context or environment within which these teams operate. Two themes emerged which appeared influence the context or environment within which software development teams exist; motivation for agility and agile outside of software development

### Agile outside of Software Development

Participants were asked to consider not only the agile working practices they carried out within the software development department but also how the practices of the wider organisation impacted on them. The extent to which the wider organisation was willing, or able, to move towards an agile way of working was found to influence how agile working practices were adopted within software development teams.

“Agile’s only done on a microscopic level at [SoftwareCorp]. It needs to be done macroscopically, and that's why whenever you read about failed scrum attempts and generally failed agile attempts it's because the whole business didn't adopt it" (Blair, PO)

Software development teams were seen to achieve some level of agility through the adoption of agile working practices. The extent of adoption was limited, and in some cases actively blocked, when it came to working with other departments. Participants described how the structure of SoftwareCorps did not lend itself to an organisation who sought to be agile. They suggested that dividing the business into functional units (sales, development, support) each with their own leader had helped to create a siloed mindset where the focus was not on end to end delivery;

“the delivery function don’t have that desire to get what the development teams build every two weeks out to the clients every two weeks” (David, PO)

Where agile existed only within the software development department, teams eventually hit 'organisational boundaries' to agile working, which subsequently created tensions between departments and their leaders, reportedly creating a "them and us" mentality.

Additionally, tensions were exacerbated by the presence of conflicting working practices with which agile had to co-exist. For example, the delivery department worked with ITIL (Information Technology Infrastructure Library), a framework which required any software changes to be approved by a change advisory board (CAB). The need to go through a CAB created delays to rolling out releases and meant slower feedback cycles. Many participants spoke of the frustration regarding obstacles inherent in an organisational structure whose parts operated independently and whose ways of working were not aligned. The implications of working within these boundaries and the resulting tensions were seen to have negative repercussions on the existing agile working practices of the software development teams;

"they're working in a very waterfall way and can't adapt to that problem we've just given them, then that might throw our whole sprint into chaos because then we're going to have to stop things and deal with them. And then at the same time, they could do something which affects us, which then breaks our way of working in an agile way…We can’t align two different ways of working” (Michael, Dev)

The customer's attitude towards agile was also a factor in the organisation's motives for working in an agile way outside of software development. The appetite of SoftwareCorps' customers to accept regular delivery of functionality, or approach projects in an agile manner, varied dramatically. Some followed traditional project management frameworks which defined how SoftwareCorps interacted with them and others were unable to accept regular releases due to significant change control processes. This resulted in releases being stockpiled and only being accepted during formally approved 'roll-out windows’. Emmett (LD) described these issues as having a “buffer effect” which constrained the agile aspirations of the organisation. He described hitting these barriers as causing “chaos in the system” and being a barrier to further agility.

In summary, the adoption of agile working practices within software development teams at SoftwareCorps was influenced by the environment in which they worked. The design and structure of the organisation, subsequent tensions and boundaries created as a consequence of this design, and the working practices of the external customers all mediated the success with which agile working practices were established within software development teams.

### Motivation for introduction of agile

An organisations' motivations for introducing agile were also found to influence how agile working practices were adopted within software development teams. This was closely related to how agile was interpreted, and as such, what the perceived benefits or drawbacks of working in this way were. The influence of perceived or stated motivations for introducing agile was particularly relevant during periods when the transition to agile working required significant effort. During these times, if there was no obvious requirement or desire to persevere, teams were seen to take the path of least resistance resulting in stagnation of agile adoption. SoftwareCorps saw several attempts at introducing 'agile' working. Participants recalled that the initial introduction of agile was motivated by a belief that working in an agile way would improve productivity and transparency of software development. It was hoped that this would address ongoing difficulties of "non-delivery, and over-sized projects which would go on for quarters, if not years, before releases would happen” (Reece, LD). This was corroborated by comments made by Richie (COO) who described his own belief that agile was a tool to ensure the organisation delivered on time, and was a way of holding people to account for their commitments;

“to me [agile] means successfully completing the task in an agreed amount of time, understanding that the chances of you having known exactly what was going to happen on day one being virtually zero…agile, it has to accommodate changes that happen and if the accommodation means everybody works a little harder or stays late, then that’s it… To me if you’re going to do it, it’s more than simply a process. It’s like I said, “I’m committed to these goals, we’ve chosen to do it in an agile way but I’ve committed to the goals” (Richie, COO)

When the results of new agile working practices did not meet expectations, buy-in and support waned. With this in mind, it was evident that how individuals interpreted agile influenced the working practices that were adopted, and that there were distinctions between those who perceived agile as something one does versus those who saw it a way of being. One lead developer suggested that some people were motivated to work in an agile way as this was what they had experience with, whereas others were simply looking for *“a way of working which is least effort for them individually”* (Reece, LD). Through the early attempts at adopting agile working practices, it was evident that where these practices were used without an appreciation the principles behind them, for why they were being used, it was easy for these practices to erode.

When considering the motivations for adopting agile, along with its perceived benefits, participants noted that agile is often promoted as a way of dealing with changing and volatile markets. They reflected, however, that SoftwareCorps did not often deal with such challenges and as such, perhaps was not *"in the sort of sector where we need to be agile"* (Joseph, PO). They also argued that the products they offered were well-established and rarely changed, and as such, their core business did not need to be agile.

In considering the differences between their early attempts at adopting agile with those later in the organisation’s lifetime, participants felt that when the organisation was establishing itself in the market, there was more of a need to be agile. However, in later years, the pressure for cash-flow was lessened and, as such, the need for agile reduced. Overall, participants felt that given the organisation's position in the market, the maturity of their core product offering, and its monetary success, the need to be agile was minimal. Consequently, the degree to which the organisation as a whole was bought in to working in an agile way was limited;

“I don’t think [agile is] something [SoftwareCorps is] particularly taking seriously. I don’t think it feels the need to. We’re quite successful as an organisation. I think success will always breed a certain amount of complacency and waste…If we were a young start-up that’s really scrapping for existence you’d look at things in a very different way.” (Adrian, SM)

Transitioning to agile working practices requires commitment from everyone involved, from the team level, throughout the organisation to those in leadership positions. Teams who are not bought-in to the idea of working in an agile way are unlikely to be moved to tackle challenges inherent in the transition, and organisations (and their leaders) who are not committed to agile working are unlikely to invest the resources needed to ensure its success.

## Factors influencing the adoption of working practices associated with Agile Principles

In Section 5.2.4, the story of how agile was introduced 'by the book' was presented along with examples of how this was done. The following section will discuss factors that were seen to influence the adoption of agile working practices. Table 16 summarises the key points from the transition to agile working and the associated factors.

Table 16: Factors influencing the adoption of agile working practices

|  |  |
| --- | --- |
| **Key Points** | **Factors** |
| Hired a CTO who was very supportive of Agile | Leadership |
| Sponsored a ‘technology transformation’ initiative to transition to new way of working | Leadership, Communication, Sponsorship |
| Held town halls to spread his vision (self-org, cross-functional, empowered teams) | Communication |
| Engaged regularly with team leads (voice, concerns, keep updated) | Communication |
| Identified existing knowledge of agile & Scrum poor | Understand current system |
| Removed existing PMO framework | Removal of barriers |
| Promoted going back to basics, do Scrum by the book | Leadership, Selection of Scrum |
| Hired an agile coach | Leadership, Coaching, Sponsorship |
| Ran training sessions on intro to agile | Coaching, Communication |
| Observed teams | Understand current system |
| Asked for a volunteer team as Pilot | Coaching, Buy-in, Piloting |
| Trained up internal SM's | Coaching |
| Coached team in roles, responsibilities and process | Coaching |
| Removal of saboteurs | Removal of barriers |

### Effective and Open Communication

Increasing communication and in particular face to face communication is a central facet of agile working. It is represented not only in the agile principles but also forms a fundamental building block of agile methodologies such as Scrum, feeding into practices such as the daily-stand-up, story writing and sprint demo. It follows, therefore, that effective and open communication was found to be an important factor in the adoption of agile working practices at SoftwareCorps.

For example, an early priority for Jamie (CTO) was to establish open lines of communication across the organisation. He did this through hosting companywide ‘townhall’ sessions to present his vision and answer questions, as well as through regular meetings with the lead developer group. In these meetings Jamie (CTO) shared updates on his efforts to drive improvements companywide and encouraged the lead developers to bring ideas, concerns and problems to his attention;

"[Jamie] started off with listening, [Zane] would never have listened, he started off with listening, and I thought well this guy is going to be great…he communicated [changes] much more personally… and he seemed like quite a friendly guy that was quite willing to have a chat" (Emmett, LD)

Effectively communicating the vision for agile working, and keeping lines of communication open in both directions during the transition are examples of how effective and open communication facilitated the adoption of agile working practices at SoftwareCorps.

#### Leaders Influence

The leader, in this case Jamie (CTO), was a key influence on the factor of effective and open communication. Jamie (CTO) led by example, embodying the agile principle of face to face communication. It was through Jamie’s approach to the agile transition, visibly sponsoring and advocating for agile working through town halls and regularly engaging face to face with team leads, as well as running organisation wide training sessions on agile concepts which differentiated this attempt at agile working from those of his predecessors.

### Coaching

In the Scrum Guide, the creators of Scrum acknowledge that Scrum is 'simple to understand' but 'difficult to master'. As such, it can be difficult for those without previous experience of Scrum and agile to succeed with it. It was found that a significant factor facilitating the adoption of agile working practices at SoftwareCorps was the provision of coaching in the form of an experienced agile coach.

This coach, Nelson, was hired specifically to work with the software development teams day-to-day in order to coach them in agile working practices, as well as help them grasp the underlying theory behind agile working. He began by running ‘introduction to agile’ workshops which covered the proposed benefits, but also the challenges that would be involved in transitioning to new ways of working.

“[Given coaching] all the teams suddenly switched on. They suddenly had somebody in a room they could go “So how should we do it? We’ve been doing it like this. Is that right?” and a person was there to say “Well, I’ve been around the sector. I’ve seen this. That worked, that didn’t. How about…” and [Nelson] was a facilitator that allowed the teams to find their own way again and find something that allowed them to do what they’re actually employed to do – make good software.” (Sheldon, LD)

This approach differed significantly from SoftwareCorps early attempts at transitioning to agile working, which involved a small number of individuals being sent on 2-day training courses. Through the support of a hands-on agile coach, teams had the tools to work through issues specific to their context in real-time. Once Nelson's initial coaching period of three months was up, teams had a good understanding not only of what the practices associated with Scrum were but why they were essential to achieving greater agility.

#### Leaders Influence

Jamie (CTO), was the driving force behind the provision of day-to-day coaching for software development teams during the transition to agile working. It was on his recommendation that Nelson was onboarded, and with his support that teams were given coaching on the ground to understand the roles, responsibilities and processes of agile working. Jamie, alongside Nelson, was also a proponent of coaching individuals internally to become scrum masters, a role which did not exist prior to his engagement. Up until this point, leaders such as Zane had sent a small number of individuals on formal training (e.g. 2-day training courses) rather than providing day-to-day coaching. Jamie’s (CTO) approach, supported by Nelson, was a significant influence on the way coaching was perceived within the organisation.

### Understanding Existing System

A fundamental principle of Kanban (a framework for process improvement often used alongside other agile methods) is to start by understanding the current system before making incremental improvements. Establishing an appreciation and respect for the existing system at SoftwareCorps was found to be a supporting factor to the agile transition.

Drawing from Kanban, Nelson spent his first weeks as an agile coach studying the existing process and structures at SoftwareCorps to gain an understanding of the organisation and how the teams worked. He described observing the culture of the organisation, witnessing powerplay and politics, which hindered collaboration, and finding that teams lacked purpose and focus.

Through establishing an understanding of the existing system, Nelson was able to take a considered and informed approach to transitioning from the established ways of working, addressing the most pressing issues first;

“My personal view was that how does one address all of these problems? Obviously, I knew the organisation has to be transformed… "this is where I want to be, how do I get there?". And to get there, there are two possible ways, in my opinion, you have to have support from the top that enables everything from the top in terms of money, time, resources. All the heavy artillery backup. From the bottom, you will have organic growth. So we picked up one team, we let that team practice and contextualise all of the Scrum and XP” (Nelson, CTO)

#### Leaders Influence

The leader, in this case Nelson, was influential to the factor of understanding the existing system as this formed part of his initial approach to supporting the transition to agile working. Based on his experience, he began by observing teams and identifying that existing knowledge of agile and Scrum were poor.

### Removal of Barriers

Taking steps to manage barriers to the new way of working was a critical factor in the adoption of agile working. Initially, the removal of the existing PMO framework demonstrated this.

"Bureaucracy had kicked in, so for you to doing anything you had to have a spreadsheet. On Fridays, we went to a meeting that people just talked about things, and they looked at the spreadsheet: red, green, amber. But that really didn't mean anything, it didn't reflect anything. So those were the things that we started with [removing]" (Nelson, CTO)

This framework presented a barrier to Jamie’s (CTO) vision for agile working as it was based on Prince2, consisting of heavy documentation, upfront designs, stonecast delivery dates and gate reviews at each stage of the waterfall-esque development process. Removal of this allowed Jamie and Nelson the freedom to redefine the process of delivering software without trying to contend with existing processes.

Another barrier encountered during the transition to agile working were individuals who were obstructive to the transition and who were seen to sabotage efforts by others to work in an agile way. Consequently, Nelson identified and removed these individuals;

“They are stopping people to deliver their best so you remove them from the mix” (Nelson, CTO)

“I had to identify whom will never be here. There is no way that they will learn… there is no way they can transition to a new way of working…all of the people that I had to part with” (Nelson, CTO)

Having the ability and willingness to remove barriers to agile working was seen to be an important factor in the adoption of agile working practices at SoftwareCorps, in particular when it came to making difficult decisions regarding personnel and established structures.

#### Leaders Influence

Both Jamie and Nelson were influential in their actions to remove barriers to the transition towards agile working. First, Jamie (CTO) led the way by immediately removing the existing PMO framework, along with it’s associated meetings and paraphernalia. This was a significant step in demonstrating his commitment to agile working, and also in creating the opportunity to change ways of working in a tangible and visible way. Second, Nelson was instrumental in addressing barriers in the form of individuals who were passively blocking or actively sabotaging the transition. Before this point, leaders had not taken such drastic actions to support the move towards agile working.

### Sponsorship & Buy-in

Sponsorship and buy-in were found to be vital in enabling the adoption of agile working practices. This was seen at both the management and grassroots level.

SoftwareCorps' management demonstrated their buy-in to the transition by hiring a CTO with a strong agile vision, and by supporting the engagement of an agile coach. They demonstrated their continued support by permitting newly agile teams the time needed to adjust to new ways of working, which they had been informed would slow teams down at first.

At the CTO level, Jamie's (CTO) sponsorship of a 'technology transformation' initiative provided momentum and support for the changes taking place. Through his townhall workshops, he promoted his sponsorship of the initiative and created a sense of curiosity across the organisation.

"He had this huge whiteboard which was explicitly the agile transformation of the company …and he did give some good presentations on it about how it was going to happen, and he did give reasonably regular updates to that" (Woody, LD)

At the grassroots level, buy-in was essential given the volunteer-driven nature of the roll-out and the significant challenges involved in transitioning to agile ways of working. Individuals demonstrated their buy-in through their willingness to be coached, by volunteering for something they were told would be difficult, and in accepting that their existing knowledge of agile might be challenged.

#### Leaders Influence

Aside from the evident sponsorship of the wider leadership team in hiring a CTO who had a strong agile vision, Jamie (CTO) and Nelson had a particular influence on the factor of sponsorship and buy-in. Jamie actively sought to garner buy-in to his vision through the townhall workshops and companywide introduction to agile sessions. His designation of the transition as a ‘technology transformation’ helped to create excitement and buy-in across the organisation. His drive to engage an agile coach, providing support for teams on the ground, also demonstrated his sponsorship and commitment to the agile approach. Jamie (CTO) and Nelson together cultivated buy-in at the grassroots level by demonstrating their own sponsorship of agile working and allowing teams to volunteer for coaching and piloting of Scrum ‘by the book’.

### Piloting with Volunteers

The transition to agile working at SoftwareCorps began with a team who volunteered for coaching and agreed to be a pilot to demonstrate how to do Scrum ‘by the book’. Piloting with volunteers was found to be a significant factor in the adoption of agile working at SoftwareCorps.

After a period of observation, Nelson moved on to actively working with software development teams to coach them in Scrum and XP. Nelson began by asking for a team to volunteer to learn how to do Scrum 'by the book'. This meant allowing him to coach them day-to-day and agreeing to follow all the 'rules' of the scrum guide explicitly for a period of 6 sprints (3 months). During the three months of coaching with the first volunteers, Nelson created what he called a 'fishbowl' environment. He ensured the benefits of how the team were working were broadcast widely and was transparent about how they were progressing. This created intrigue from the other teams and resulted in more volunteers coming forward to request training and coaching for themselves. All teams were allowed to decide NOT to continue with Scrum if they found it had not been successful for them, although none did.

“[Jamie and Nelson] said “open offer, we have assistance here. If you’d like to learn more about this and become more scrum there is a mentor” and those [who volunteered] were easier to convert…those people loved it, absolutely amazing responses from their sessions. They came out really flying, walking on air.”

Establishing an environment where teams stepped forward to 'pull' coaching, rather than having coaching 'pushed' onto them, empowered teams. This meant they were open to changing their working practices, resulting in a successful pilot which effectively demonstrated the benefits of agile working to the wider department. Piloting with volunteers, therefore, was found to be a supporting factor in the adoption of agile working practices.

#### Leaders Influence

Nelson was the driving force behind creating a pilot, demonstrating agile working with a team of volunteers. It was his decision to approach the coaching of teams in this way, and he focused on making the ways of working during the pilot visible across the organisation.

### Selecting Scrum

The decision to coach teams in agile working through the use of Scrum was found to be a key factor in the adoption of agile working. The Scrum Guide, which outlines the roles, events, artefacts and rules which define Scrum, is based on a set of values (commitment, courage, focus, openness, and respect), and is founded on empirical process control theory with principles of transparency, inspection, and adaption. It defines a scrum team as self-organising and cross-functional, delivering products iteratively and incrementally to maximise opportunities for feedback, and ensuring a potentially useful version of the working product is always available. As co-creators of the agile manifesto, the founders of Scrum helped shape the agile values and principles based on their experiences using 'lightweight' methods focused on improving software development. As such, the adoption of Scrum as a framework facilitated the adoption of agile working practices in teams at SoftwareCorps.

Teams were first taught to do scrum 'by the book'. This enabled Nelson to ensure teams consisted of a dedicated product owner and scrum master in addition to the software development team. In the pilot team, this involved addressing the poor relationship between the development and product teams. Through adhering scrum, Nelson established the norm that each scrum team must have a product owner and scrum master embedded within the team, collaborating and working effectively together, set the example and expectation for Scrum subsequent teams.

“So when [Jamie], shortly followed by [Nelson], first came in they did just tear it all up and just say ‘we’re going to do this by the book’“ (Woody, LD)

Initially learning Scrum 'by the book' had two benefits. First, it ensured that individuals experienced Scrum and its practices as they were intended to be carried out. Second, it enabled them to learn the values of Scrum and understand the purpose of working practices in this context. Without this, teams risked adopting a 'Scrum-but' version of Scrum where some of the core values are missed, and practices are selectively adopted, as was seen to happen in earlier attempts at agile working at SoftwareCorps.

#### Leaders Influence

It was the decision of the leaders, specifically Jamie and Nelson, to begin the transition to agile working by following Scrum ‘by the book’. It was their drive to take SoftwareCorps ‘back to basics’ by following a well-defined method which influenced this factor.

## Leadership and Agile Adoption

Leadership emerged as a theme across many of the factors discussed in this chapter. As such is presented as a critical factor in the adoption of agile working practices at SoftwareCorps.

Most notably, leadership emerged as a critical factor through the impact of leadership changes on agile working practices. An increase in agile working practices were seen when Jamie, and shortly after Nelson, joined SoftwareCorps. However, when Julius later became CTO, there were significant changes in how the technology department operated. In particular, there was a lack of support at the CTO level for agile working. This lack of support ultimately led to what participants called the 'undoing' of agile and a 'return to command and control'.

Not only was leadership an independent factor, it also appeared as a theme throughout the other factors presented, as described throughout Section 6.15. For example; CTO sponsorship of an agile transformation, the businesses buy-in to hiring an agile coach, leadership provided by agile coaching, leaders clearly communicating the agile vision, and leaders empowering teams to own their way of working through volunteering to be coached in agile practices.

For example, Nelson was responsible for choosing to use the Scrum methodology as a vehicle to train teams in agile working practices. He was also responsible for defining how this would be rolled out, via introductory workshops, volunteering and creating a 'fishbowl' environment to broadcast his methods. He also provided agile leadership daily by personally coaching each of the teams for their first three months using Scrum.

Given its emergence as both a critical factor and a theme within other key factors, leadership is explored in more detail in Chapter 7.

## Summary of Phase One Findings

To understand how, and to what extent, agile principles were adopted, focus was given to the working practices associated with each principle. The mapping of practices to principles developed in Table 5 (Section 2.3.2) from the literature was applied to the case under study and an assessment of how this reflected adoption was presented.

The adoption of practices at SoftwareCorps were referred to as either behavioural or cognitive; a behaviourally adopted practice is carried out in action but does not form part of the individual's values or belief system. A cognitively adopted practice describes actions which are being carried out because the individual's values and beliefs are aligned with the consequences of actions associated with these practices. These terms were used to demonstrate the distinction which was found to be evident in the way different teams and individuals carried out agile working practices at SoftwareCorps.

The adoption of agile working practices at SoftwareCorps was found to be influenced by the context in which the development teams worked. The design of the organisation, boundaries created as a consequence of this design, and the working practices of customers were all found to mediate the success with which agile working practices were adopted within software development teams.

Additionally, transitioning to agile working practices was found to require commitment from everyone involved, from grassroots to the highest levels of leadership. Teams who were not bought-in to the idea of working in an agile way were unlikely to be moved to tackle challenges inherent in the transition, and leaders who were not committed to agile working were unlikely to invest the resources needed to ensure its success.

Through the analysis of each of the 12 agile principles and associated working practices, it is clear that there are a variety of factors which influence the adoption of agile working. These include effective and open communication, coaching, understanding the existing system of working, removal of barriers, sponsorship and buy-in, piloting with volunteers, selection of Scrum, and leadership. However, it was evident that leadership emerged as the most significant influence in the case under study.

Leadership began to emerge as a key theme through both participant interviews (all 27 participants from phase one referenced leadership in some way) and researcher observations. In particular, the way the software development department operated appeared to be largely dependent on the steer given from the individual in the role of CTO, as one scrum master reflected;

*"I mean if you look at any organisational behaviour textbook it'll say everything comes from the top down. So yeah, you're beginning to see… Obviously, the new CTO comes in and wants to try and put his own ways of doing things and often whenever people try to implement something they go "previously this really worked for me." (Paul, SM)*

Participants associated different ways of working with specific individuals. The researcher was also able to observe changes that took place when a new individual was introduced into the CTO role. Through the participants’ descriptions of different phases of working and subsequently their accounts of individuals in leadership positions, distinct styles of leadership emerged, as will be described in Chapter 7.

Subsequently, variations between how agile was enacted under different leaders became apparent, for example, team size and team management. Although most leaders called their way of working agile, there were clear differences between working practices carried out as well as in the style of leadership. In the following chapter findings will be presented regarding the impact of leadership, leadership style and a leader's interpretation of agile on the adoption of agile working practices.

# Phase Two Findings - Leadership

This chapter presents the findings regarding the influence of leadership on the adoption of agile working practices and principles. In so doing it addresses *RQ2: How does leadership style affect the adoption of agile working practices in software development teams?* Data collection for phase two of this study took place during the “undoing of agile” (2017) period illustrated in Chapter 5, Figure 7. This period commenced with the resignation of Nelson, who was replaced as CTO by Julius. This period coincides with the sale of SoftwareCorps to a new parent company and significant uncertainty surrounding how the business would be structured going forward.

An analysis of two leaders’ influence on the adoption of agile working practices at SoftwareCorps will be provided. This analysis will be done by examining the influence of each leader in turn on the adoption of two agile principles; Self-organising Teams and Reflect and Improve. First, leadership will be explored through the participants’ perceptions of the two CTO's who held positions during the data collection phases of this study. Both their style of leadership and their interpretation of agile will be explored. Second, the relationship between leadership and the principles of Self-organisation and Reflect and Improve will be explored.

The conceptual model of leadership and agile adoption is provided below (see Figure 12) which will emerge from the findings described later in this chapter. It is presented here to provide a reference point for the reader, as each piece of the model is built up throughout the chapter. A full discussion of the model is then presented in Chapter 8 (specifically in Section 8.4).

Figure 12: Conceptual Model of Leadership for Agile adoptionA screenshot of a cell phone

Description automatically generated

## Leadership

In this section, the leadership styles of two leaders, Nelson and Julius, will be explored in depth. These two leaders were chosen as the focus for this study due to the notable influence they had on agile working during the period of data collection. Their actions as leaders and respective interpretations of agile and agile working practices will be presented, with illustrative examples.

### Nelson’s Leadership Style

Nelson defined his leadership style as a *“meritocracy and cooperative model*[[6]](#footnote-6)*”,* based on his self-described Gestalt philosophical perspective that *the whole is other than the sum of its parts*;

*“I have the fundamental belief…that an outcome of a groups work, regardless of how mediocre they are, it is far more superior than the outcome of a single genius because single genius is susceptible to be run over by a bus. The group, however mediocre… their outcome is sustainable. That’s why I genuinely believe in the German school of philosophy in terms of Gestalt, and genuinely because I have a more social democrat leniency” (Nelson, CTO)*

Nelson also noted that, to him, the word ‘leadership’ was misleading. He preferred instead to describe his role as *“facilitator, coordinator, enabler”,* emphasising his ultimate goal to enable individuals to deliver value.

Having joined the company as an agile coach, before transitioning into his role as CTO, Nelson had a reputation as an agile evangelist. One scrum master, Frankie, described him as a *“super agile man”,* and as someone who believed in agile *“fully to the letter”* with a heavy focus on learning for both individuals and teams. Frankie described Nelson’s style as giving *“power back to the people who would be doing the work”,* allowing those closest to the work to define how it should be done.Nelson was described by Taylor (DiT) as being articulate and passionate, with an awareness of body language and linguistics, and being skilled at communicating with different types of people. On describing his communication style, Nelson described himself as;

"Direct, passionate, but honest to a point of bordering with difficult. Honest to a point where you say nothing to a person's face that you would not say it behind their back" (Nelson, CTO)

The scrum masters, who had been coached by Nelson, felt that his leadership style was one which embodied the philosophy of agile. Devolving responsibility of ‘doing’ to those that ‘do’ the work. The scrum master group reportedly felt both empowered and supported by their leader to do what they did best – drive improvements across their teams;

*“There was an expectation that people do sort of step up, take responsibility, decide on problems, sort things out…which I think was really positive. I mean obviously I enjoyed working very much under that” (Adrian, SM)*

Conversely, some participants suggested that their experience of Nelson varied based on how closely aligned to his objectives and views they were;

*“Nelson’s leadership style was very much my way or the highway, and for many it resulted in the highway, but it was also a charm offensive as well. So, the people who he liked…he almost rewarded people who he considered allies” (Blair, PO)*

Nelson’s leadership was put to the test during the transition towards agile ways of working. During this transition, there was friction caused by members of the software department who were resistant to the proposed changes. Adrian (SM) reflected that these individuals appeared to object to a perceived loss of power, prestige, or position. For example, when asked to make their work visible on the team scrum board, to aid with transparency across the team, one individual is reported by Nelson as responding; *“over my dead body will I write notes of what I will be doing today!”*.

Similarly, Laurie (LD) recalled how there were individuals who did not want to do agile, as they were comfortable with their existing ways of working. He also suggested that the more senior team members felt that what was being asked of them was *“beneath them”*. For example, when it was suggested that the whole team were responsible for quality and testing, Laurie (LD) recalled a lead developer who objected that testing was not part of his role*.*

Reflecting on this period, Nelson (CTO) recalled what he described as *“power centres”* of people. These power centres were actively against agile working practices, and Nelson feared these individuals would “*immediately go back to the old-style"* of working if left to their own devices, influencing others to do the same. He recalled how, during his early engagement with teams, he recommended they use Scrum. However, these influential people would claim to be following any process other than Scrum, such as Kanban or DSDM, as a way to create conflict. Nelson spoke of *“regrettable”* leadership decisions which he had to take regarding individuals who, he believed, would be unable and unwilling to transition to a new way of working, resulting in several people leaving;

*“So a lot of people then were forced to be subversive and… Well, we know they got pushed rather than side-lined. So, they were now getting pushed towards the exit” (Blair, PO)*

For some participants, Nelson's actions in removing people suggested he did not always practise what he preached; as a leader who publicly championed self-organisation and supported the empowerment of individuals, it appeared contradictory that he should remove anyone who did not support his views;

*"I felt like everyone was just a piece on a chessboard that he was pushing around, and no one had any opinion. Even though he was stressing that you can express your opinions, but as soon as you disagreed or suggested something different he dismissed it, if he didn’t think that that was a good idea, and it turned against you” (Taylor, DiT)*

Some participants referred to Nelson as *“an assassin”* due to his tendency to 'get rid' of people (from the organisation) who were not on board with his vision. “Assassin” was a label Nelson used himself when describing how he felt when having to make these decisions.

Gabriel (DIT) was a member of a team where several members had left as a consequence of intervention by Nelson. He described how this had been effective at removing protestors of agile, although he questioned the organisational impact;

*“I mean I think it’s fair to say there was a problem with individuals just not wanting to change the way they were working, and in that case perhaps just getting rid of people is the best way to do it …it feels like that was really the solution that worked for [us], which is to say that, the way that the change happened was just by replacing people, which frankly doesn’t sound like the best way of doing things” (Gabriel, DIT)*

Most participants described Nelson's strength as his in-depth knowledge and experience as an agile coach and trainer. In their words, he *“knew the theory”* however there were concerns about his experience as a *“technologist”*. Richie (C-level) suggested that although Nelson was strong when it came to people, relationships, and agile, his technical skills were left wanting. This perception led to views by many that the software department was strong in agile working, but that technology had fallen by the wayside.

Emmett (LD) recalled how Nelson acknowledged that he was not an expert in SoftwareCorps systems, and would consequently encourage those who were experts to make necessary decisions on design or technology. This approach was something Emmett (LD) saw as a *“ray of light”* in comparison to a previous executive manager, Zane, whom Emmett (LD) felt lacked understanding but would make such decisions on behalf of the developers regardless. He shared concerns, however, that a lack of understanding of SoftwareCorps’ systems undermined Nelson’s influence with the executive management team;

*“[knowledge of our systems] would have been really useful to him up there... because sometimes I think you need the people higher up in the company to be able to argue to get stuff done”* (Emmett, LD)

This concern was validated to some extent by Richie (C-level) who commented that Nelson’s *“technology skills at some level weren’t up to it”.* Richie suggested that Nelson left as a result of the arrival of the new owners who, he believed, would be involved and capable of assessing technology decisions.

In summary, Nelson's leadership style is best described as 'transformational', with a focus on empowering individuals and encouraging a focus on team and organisational goals, rather than individual desires. However, he adopted a more transactional approach in certain circumstances, in particular when put under pressure. In the following section, Nelson’s interpretation of agile will be described and how this influenced how teams worked will be presented.

### Nelson’s interpretation of agile and resulting impact on working practices

Nelson described agile as *“high-frequency working and anything else that comes with it”.* He believed agile was the *“purposeful assembly of people to deliver a shared goal”.* His definition of agile and the way he believed it should be implemented focused heavily on the human side of agile, such as empowering those who do the work to define the work, creating a purposeful environment, and building a *“learning company”*.

Under Nelson, team size was based on the scrum guides recommendation of between 3 and 9 people. Participants noted that these teams were large enough that *"good-sized pieces of work could get done"*. The most recent version of the Scrum Guide (2017) available at the time of writing states that;

*“Optimal Development Team size is small enough to remain nimble and large enough to complete significant work within a Sprint. Fewer than three Development Team members decrease interaction and results in smaller productivity gains. Having more than nine members requires too much coordination. Large Development Teams generate too much complexity for an empirical process to be useful”* (Schwaber & Sutherland, 2017, p. 7)

Nelson also focused on creating cross-functional teams of multi-disciplined people, referred to as ‘T-shaped’ individuals (e.g. Demirkan & Spohrer, 2015). T-shaped individuals are said to have an in-depth knowledge of a core competency, as well as broad knowledge in other areas. Multi-disciplined individuals allowed teams to organise around the work rather than organise the work around the people, and built-in more resiliency to holidays and sickness. An example of this in practice was encouraging ownership of quality and testing by the entire team;

*“[Under Nelson] I was… one of about five or six developers with one manual tester, essentially. So, whilst we had a manual tester on that team, she wasn’t the only one who was doing testing. We were all doing manual testing and automated testing as developers as well”* (Quinn, Dev)

In order to formalize this approach, Nelson revised the job titles in the software department so that everyone was called 'developer', effectively removing the title of 'tester'. Quinn (Dev) described how this sought to achieve a reduction in *“throwing things over the wall to a tester”,* and believed this objective had been met. As a result, developers were more conscious of testing. Quinn (Dev) felt, however, that the approach was too extreme, suggesting that *"to deny that there is a difference in speciality is a mistake"*. This message was echoed by Laurie (LD), who noted that as a consequence of the title change senior members of the testing team, himself included, felt reduced to junior members of the software development team. This negatively impacted on their motivation. Michael (Dev) also struggled to understand how individuals who could not code could be considered developers;

*“I mean the testing one is an interesting one as well because there are some people who actually are unable to write code, so generally just do testing, which even though they’re a ‘software developer’ I’m not quite sure how that works”* (Michael, Dev)

Conversely, some saw the benefit and understood the intention behind what was being proposed. David (PO) explained that, in his experience, a broader skillset was required for agile to allow teams the flexibility and capacity to organise around the work to be done. Under Nelson’s model of streamlined job titles, there was still an opportunity for specialism, with team members working to their strengths where possible. However, the expectation was that individuals supported other roles when needed. Emmett (LD) believed the removal of the ‘QA’ and ‘testing’ title had contributed to the elimination of the term ‘QA backlog’. This term referred to a queue of testing waiting to be completed, a problem which had plagued teams in the past and vastly reduced the speed at which they could deliver. A QA backlog would exist where a team had a ‘lack of QA’ to carry out testing, but where developers would continue to write code which added to the queue. As many developers did not consider testing a part of their role, and there were always more developers than testers, delivery of products would be throttled by the capacity of the QA team to process the queue of work. This splitting of responsibility also shifted accountability for getting a piece of functionality to customers to the QA team. Consequently, a popular reason given for delivery dates slipping was a ‘lack of QA’ and the phrase 'its code complete' was often used. By promoting the agile principle of creating cross-functional teams and supporting individuals in becoming multi-disciplined, Nelson was seen to address these issues.

Nelson also focused on increasing transparency and information radiation across the department and the wider business. He implemented company-wide end of sprint demo’s, called ‘sprint snapshots’, which were held at 9.15am, every two weeks. This event was in line with the two-week sprint cadence to which all teams were working. The demos were carried out in a large communal space near the office kitchen and were promoted as a convenient way to engage with the software development teams. Attendees were encouraged to bring their breakfast along, to create a comfortable and informal environment in which information could be shared. Each software development team had a 5-minute slot to showcase what they had built in the latest sprint, as well as sharing what would be coming in the next sprint. Taylor (DIT) described this as Nelson’s way of increasing transparency and keeping the business informed.In the past, teams had held the end of sprint demos independently, in small meeting rooms. Often each team’s (of which there were eight or more) demo lasted 30 minutes and was only attended by the team themselves. Reflecting on this, Reece (TL) remarked on the significant improvements realised under the new format, and the resulting impact on collaboration across the business;

*“now, it’s actually educational and people actually want to turn up from the other side of the business to see it, which, I think, has improved the general kind of interaction and collaboration between technology and other parts of the business*” (Reece, TL)

Another key practice to increase transparency was the use of physical team scrum boards and whiteboards around the office, something which Nelson was a particular advocate of;

*“the first thing that I observed was that [the] board was not central radiator of information. It was a place for people to gather around, it was a campfire without the fire itself. [You could have] put a cross there saying “gather around there”, rather than being a campfire that everyone sits around it, and there is a warmth, there is a light” (Nelson, CTO)*

Through Nelson’s coaching, not only did every team have a physical scrum board but whiteboards began to be used for other purposes and in other departments.

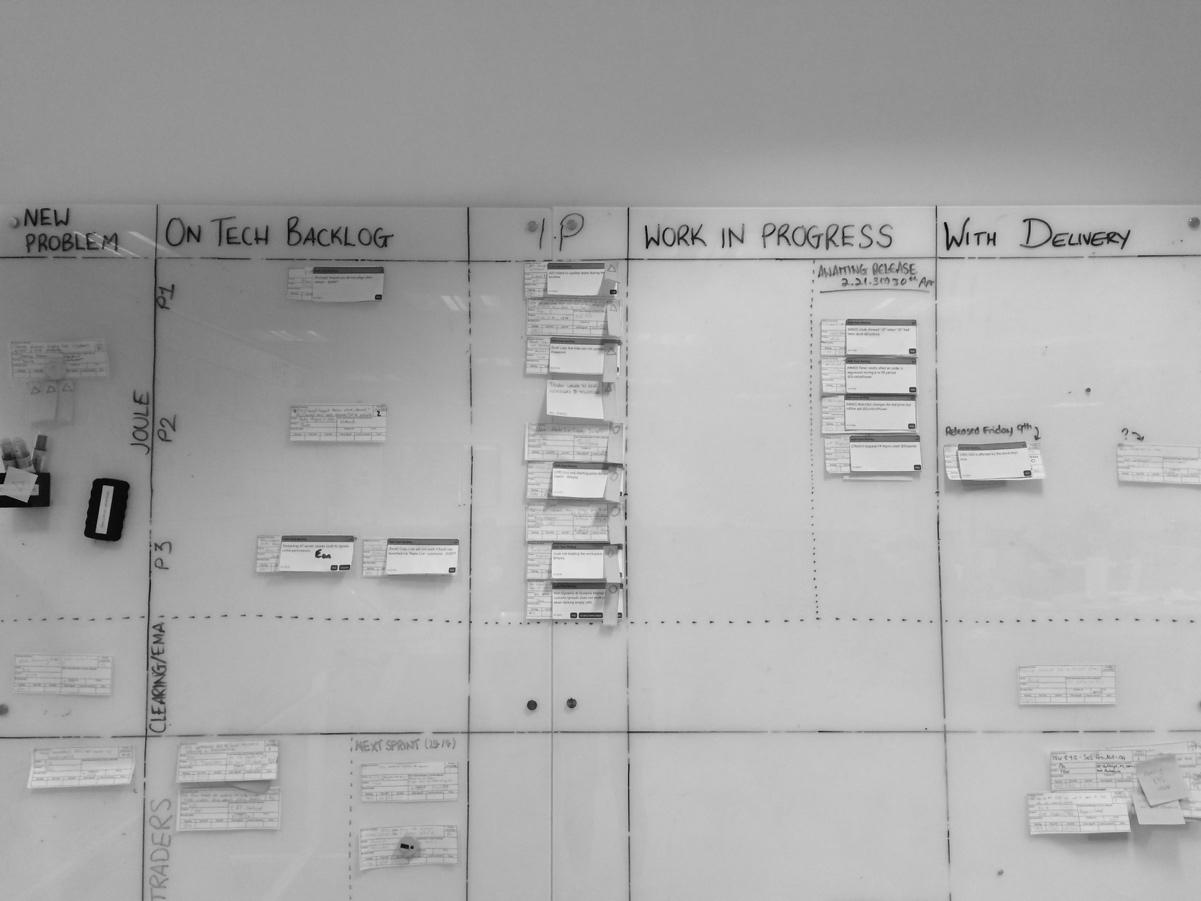
Under Nelson's leadership, there were a number of initiatives designed to increase transparency and information sharing across the business. Leading by example, Nelson created a centralised view of the projects being worked on across all software development teams on a whiteboard. This board made the project lifecycle and progress of each project through it visible. Projects moved through six phases; Identify (project idea proposed), Mobilise (gathering high-level requirements), Build (in development), Deploy (handed over to delivery team), Launch (live in production) and Close (project completed). The board was visible to the entire technology department and was located in a walkway where those passing by could easily stop to look at it (see Figure 13). Various people from across the business, such as Product Managers, C-level executives, and Sales and Client Managers, were often seen gathering around the board to discuss priorities and mitigation of potential risks.

Figure 13: Roadmap Board

An additional example of whiteboard usage was the creation of a ‘problem management board’ to address a lack of communication and information sharing between the software development department and the client-facing delivery department. On this board (see Figure 14), client issues that could not be resolved without intervention from a software development team were raised by first-line support. A daily stand-up took place around this board, where the most pressing customer issues were prioritised, and owners were assigned in both departments. Updates were then given on the progress and state of each issue to the group. Sub-groups would meet after the session to cover anything requiring more in-depth discussion. Frankie (SM) likened this meeting to the scrum of scrums, a technique to scale Scrum, where ‘ambassadors’ from each scrum team come together to raise impediments and blockers, aimed at improving coordination between teams. In Frankie's (SM) view, the key to the success of the problem management stand-up was that the people actually doing the work would *“come together to…decide how they were going to approach that work”.*

Previously, customer issues requiring investigation by people outside of the delivery team would slip through the cracks due to a lack of communication between the two departments. This was exacerbated by a lack of visibility between the tracking systems the two departments used - Team Foundation Server in the technology department and Dynamics in the delivery department. By encouraging the usage of physical whiteboards as information radiators, and regular face to face meetings between those responsible for the work, Nelson increased transparency and encouraged more effective communication.

*“[At] the problem management stand-up…every day the [delivery] team’s consultants and the product owner and the scrum master and maybe a developer from each team would come together to look at the individual bugs or issues that have come in the day before, assign them a priority and then decide how they were going to approach that work. So, the good thing that used to happen with that is that you never missed an issue, you always knew where you were and you could get a quick turnaround time if it was necessary” (Frankie, SM)*

Figure 14: Problem Management Board

In summary, Nelson’s interpretation of agile focused on empowering individuals and allowing those who did the work to decide how it should be done. It also led Nelson to advocated for the creation of scrum teams small enough to be nimble, but large enough to complete significant work, and to create cross-functional teams of T-shaped individuals to allow teams to organise around the work to be done. Finally, Nelson’s interpretation of agile demonstrated a focus on transparency and information radiation in order to improve collaboration across the business.

### Julius’ Leadership Style

Julius had worked in SoftwareCorps’ domain for many years before joining the company, and was something of a subject matter expert in his field, describing himself as having both customer-facing experience and strong technical expertise. Participants described Julius as knowing *“what he wants”* and *“when he wants it”,* which applied not only to his strong opinions on technology choice and usage but also to customer requirements. One PO, Blair, suggested that Julius *“probably believes that he knows what [customers] need over what they want, in a Steve Jobs-esque style, complete with turtleneck”.* On describing his own beliefs and values, Julius stated that communication, honesty and transparency were key to the way he managed the department. He described that, wherever possible, he openly communicated the reasons for his decisions;

*“I have reasons for the things I’m doing, for the decisions I take. I’m not ashamed for the decisions I’m taking, therefore I’m openly communicating them as much as I’m allowed to” (Julius, CTO)*

He spoke of the importance of communicating his thinking and motivations so that others could appreciate and understand why any given decision was made. This was an approach he encouraged others in his charge, such as the lead developers, to take. Reflecting on his own leadership style, Julius acknowledged that he is *“not very skilled”* when it comes to emotional intelligence. However, he believed that he compensated for this though “*a good understanding of logical cause-result relationships”,* and described his style as *“blunt, but honest and…consistent”.* Julius also acknowledged that he felt more comfortable managing people who worked within his sphere of expertise, as this would allow him to set expectations based on his own experience. He would benchmark his expectations by assessing the seniority and skill of an individual;

*“[For] something that I’ve done myself [before] then I can [manage] it based on my own experience. Based on my own judgement I can then set timelines for my expectations based on trying to remember at a given level how fast I would be able to do it” (Julius, CTO)*

Julius was well respected for his technical knowledge, by both developers and the wider business. For example, Luke (Dev) appreciated Julius’ precise, efficient approach and strong domain knowledge. It was an approach Luke (Dev) highly respected and admired. One executive stated that *“Julius could … do everybody’s job in about a third of the time”* and this sentiment was evident in the approach Julius took in his role as CTO. It was also evident in Julius's description of preferring to manage people in roles he had himself carried out. Overall, Julius had a reputation as a strong and experienced technician, with a proven record of delivering on commitments. For some, Julius’ clear and direct approach was ideal for the role of CTO. Gabriel (DIT), for example, described how he felt Julius was able to steer the department in the right direction in terms of long-term objectives, given his experience and strong industry knowledge. He also highlighted the advantage of Julius’ understanding the architecture of SoftwareCorps’ technologies, which he felt gave Julius an edge when it came to effectively prioritising work. Having a strong technician leading the department was something many participants cited as an advantage, with developers including Luke speaking of a desire to learn from and be mentored by him. Luke (Dev), however, accepted that time for this was limited since Julius had become CTO.

For some, however, Julius' intelligence and strong technical capability did not compensate for areas in which they felt he was weak. For example, Darren (Dev) questioned the effectiveness of a leader who lacked social skills. He described working with Julius as “*like working with an* *automaton”* and spoke of his discomfort during meetings due to Julius’ intensity. Blair (PO) agreed with these sentiments, declaring that Julius’ people skills were effectively *“non-existent”*. Participants described Julius as dictatorialdue to his tendency to want things done his way*,* and hierarchical in that he positioned himself such that information and decisions had to pass through him first;

*“[He’s] the kind of guy that gets all of the problems come to him…he will sit there and try and solve it in his own head first, and then he’ll look round and the first person he sees will get a “Can you do this?”. But it’s not the nice pieces of work, it’s always the stuff that he doesn’t want to do” (Frankie, SM)*

This hands-on style was a recurring theme in across participants’ descriptions of Julius’ approach. There were accounts of situations where Julius would intervene *“when it’s not going well”* to show individuals how things should be done, often in the form of coding something himself. Adrian (SM) spoke of situations where Julius would re-write developers code and check-in changes without following standard processes, such as code reviews and pull requests. It was assumed that, given Julius’ seniority and knowledge of the platform, he felt above following these processes.Adrian (SM) also described how he had seen Julius give individuals one or two opportunities to do things his way before he would step in and do it himself. An example of this intervention was observed when a feature was developed on a platform Julius had worked heavily on. Despite no longer being the lead developer for the platform, Julius continued to review check-ins for even in his role as CTO. Having identified a change he was unhappy with, he re-wrote the feature. Julius subsequently walked the lead developers through how he believed the feature should have been written in order to coach them in best practice.

Participants indicated that this behaviour was expected from Julius, especially by those who had worked directly with him when he was a lead developer. Adrian (SM) commented that *“people are always bowled over when I talk of our CTO writing code and checking stuff in”* however, the developers at SoftwareCorps appeared to have accepted this approach. Adrian (SM) suggested that teams would agree to whatever Julius suggested, purely to avoid *“tiresome”* arguments. In addition to providing strong direction in terms of what should be done, Julius also set clear expectations regarding how long a piece of work should take, as Ted (LD) recalled;

*“he has specific thinking on… timescales, and when they don’t match his timeframes he largely ignores the team and just dictates what’s going to happen, without trying to solicit an understanding of why those are the timeframes, or what’s in the codebase that makes it such” (Ted, LD)*

There were cases where Julius set timeframes for teams working on legacy platforms that they felt were not realistic, given the constraints they were working within. Namely, very old technologies and fragile architectures. Darren (Dev) suggested that Julius did not appreciate the complexity of working on legacy products. He recalled the frustration felt by the team in being told how and when to do things by someone who, they believed, had *“only really worked on relatively new products that go to a relatively small number of people compared to our software”.* Darren (Dev) also explained that if his team questioned or attempted to change Julius’ timelines, they were met with anger and pressure to align themselves with his expectations. This response often led to conflict and a general feeling of disempowerment;

*"When we look at what he's proposing and conclude that it cannot be done, at that point he will come down and get cross and tear apart all of our plans and try to find ways to make it fit his grand design… It's a matter of "This is your job and you will do it and you will meet my expectations," which is not brilliant, to be honest... At my level he's not very interested in my buy-in beyond this is your job and you will do it." (Darren, Dev)*

In summary, Julius' leadership style is best described as transactional. He was seen to monitor an individual's performance actively and take corrective action if deviations occurred. He was also seen to use contingent reward, where individuals were set explicit performance goals for which they would be rewarded. In the following section, Julius’ interpretation of agile will be described and how this influenced how teams worked will be presented.

### Julius’ interpretation of agile and resulting impact on working practices

When defining agile, Julius focused primarily on the mechanics and process of agile, while emphasising the importance of managing time and scope for project delivery;

*“Agile means we have a process that’s built on the assumption that requirements change over time…at the core agile tries to deliver stuff at a given timescale with a, to some degree, variable scope. But it's not completely open. So, the goal is always to at least follow the Pareto principle*[[7]](#footnote-7) *that you provide 80% of the functionality.” (Julius, CTO)*

Julius described himself as preferring to pick *“the right tool for the right job”* rather than being attached to any particular process. However, there were some areas where he had specific thinking about how things should be done in terms of working practices. These included optimal team size, team organisation in terms of specialisation and cross-functionality, and customer and domain knowledge.

Julius was instrumental in creating what he called small *“efficient”* teams, a construct Paul (SM) suggested stemmed from Julius’ background in consultancy where this format was commonplace. Participants coined these *“micro-teams*”, due to their size, and reported that teams felt fragmented and less empowered in comparison to working in larger teams*.* As a consequence of splitting people into micro-teams, the number of teams overall increased. Each of these micro teams was charged with a specific objective, which led to the perception that more work was in progress. However, Blair (PO) suggested that that the progress of each of those things would theoretically be slower, as fewer people could work on them. One developer, Luke, noted that he was in a ‘team’ of one. He felt uncomfortable seeking advice from those on other teams as he did not want to distract them from their projects. Adrian (SM) argued that *“nobody wants to be working on teams of one or two. It’s not motivating, it’s not enjoyable… because you can’t get an effective amount of work through two people”.* Michael (Dev) also noted that the *“sense of team”* had vanished and that he was unsure with which team he should identify. From Julius' perspective, creating small teams and assigning them small pieces of work was a way of optimising the work being done and maximising efficiency;

*“I try to promote smaller parts of the team working on items because most often this is more efficient… different tasks have different service areas, meaning only a distinct amount of people can work on something at the same time without interfering too much with each other.” (Julius, CTO)*

Julius also believed that individuals should focus on their area of expertise, and specialise to become an expert in their chosen area[[8]](#footnote-8). He saw this as a way to maximise efficiency, as people only worked on things they were very good at, and as a way to provide clear development opportunities for individuals. Commenting on Julius’ preference for subject matter experts, Ted (LD) noted that this meant teams were largely unchanging and that they were focused on a specific product. Teams were only seen to change when a specific product need was perceived by Julius, at which point he would move people around. Ted (LD) argued that this created more of a dictatorship, with minimal opportunity for individuals to come up with *“the best way of working”* for themselves. Under this model, Quinn (Dev) described how QA was relied upon to do testing, while developers focused on coding and implementing stories. Frankie (SM) corroborated this and highlighted that, as a consequence, there was a more significant distinction between the two roles. Frankie (SM) also noted a tendency for management to emphasise who the *“best”* developers or testers were to carry out a specific task. This tendency reduced the opportunity for the team to decide for themselves.

On the subject of cross-functional teams, it was Julius’ view that these made sense in some cases,but in other cases, it may not be necessary. Julius' position was that cross-functionality should be determined by how much additional productivity it would bring. In one example, of a green-field project running on SoftwareCorps own infrastructure, he saw the benefits of creating a cross-functional team to accelerate delivery. However, he suggested that for systems such as SoftwareCorps front-end product, which was deployed to a large and disparate customer base, the benefits were limited;

“…like our front end. In this case, there is hardly a benefit if people are working closer together because the problems that are occurring will always be punctual. They will never be representative, you can only see trends, and this doesn't involve our own infrastructure et cetera, so there is less benefit gained” (Julius, CTO)

Where SoftwareCorps had in the past attempted to create cross-functional teams for these products, Julius’ experience had been that the benefits were not clear and that prioritisation channels were unduly influenced based on which customers a particular operations team member was responsible for;

*“You end up with a situation where you are potentially over prioritising the customers for which the operations guy in the team is responsible for, compared to other customers for which other operations people are responsible for, which are not represented in the team. And you can't represent them all because there's just far too many. This is an area that doesn't work too well” (Julius, CTO)*

Julius noted that when considering creating cross-functional teams in the past, instead of focusing on creating efficiencies, there had been debate regarding which department *"dominated or ruled"* the team. He suggested that the wider business only agreed to cross-functional teams on the premise that their representative would be in charge, and being unable to resolve this conflict cross-functional teams were never established.

One aspect of agile working Julius spoke especially passionately about was getting developers closer to the customer, increasing the amount of face-to-face communication they had with real users. At SoftwareCorps, traditionally, developers had no direct contact with customers. This interaction was reserved for sales and client managers who would feed requirements to the product team. The product team would, in turn, communicate with the software development team. Julius felt that this caused communication to be *“muddled”* and believed it limited the chances of developers receiving valuable feedback from end-users;

*“[The message is] subjective based on what people understood, and each layer of communication also adds some indirection, so the developers were never actually exposed to the customers and never got direct feedback from them.” (Julius, CTO)*

He felt there was a disconnect between how the developers understood the products being built and how customers actually used them. In order to overcome this, Julius facilitated workshops where customers were invited to SoftwareCorps’ offices to meet the software development team. In these workshops, customers would articulate the issues or requests they had for the product to the developers directly. This allowed the developers to ask questions and understand what the customers needed for themselves. These workshops were feasible where the customer group was relatively small but was more problematic for larger customer bases. In these cases, Julius instigated client visits where developers would join client managers at the customer’s site to see how the products were used in the real world.

Will (PO) believed that increasing customer exposure for developers had resulted in an increased understanding of the products and domain, recalling how impressed he had been at how much individuals had learnt through these visits. On his team, he noted that developers would write up the outcomes of any visits to share with the rest of the team. Will (PO) felt that this had been successful in *“enticing them into caring”* about the background of customers and requirements. He credited Julius (CTO) with promoting the importance of understanding customers and the wider domain and was very supportive of achieving this through increased contact with customers. Some participants, however, questioned whether developers would be welcome when visiting customers. Emmett (LD) joked that client managers would feel like they were *“dragging these weirdo’s round with [them] and they’re just going to be really quiet and not really see much of the background anyway”.*

In summary, Julius interpreted agile as a process used to mitigate the risks inherent in product delivery. In particular, the risk that requirements are likely to change. Given this view, he believed in selecting the appropriate tool for the job rather than adhering to a specific agile method.

### Summary of Leadership Theme

Leadership was found to be a theme in all 27 phase one interviews, which led to a further phase of data collection and interviews focusing on leadership. Three key points were found through the analysis of this data; changes in leadership, different styles of leadership, and leader’s interpretation of agile were all found to impact on the adoption of agile working practices at SoftwareCorps.

In summary, Julius replaced Nelson in the role of CTO in 2016 and, as can be seen in Table 17, there were differences both in terms of their styles of leadership, interpretation of agile, and in how they influenced working practices at SoftwareCorps. These factors inform RQ2, helping to explain how leadership influences the adoption of agile working practices.

Table 17: Comparison of CTO's Nelson and Julius

|  |  |  |
| --- | --- | --- |
| Style attributes | Nelson | Julius |
| Team management and organisation | *Scrum teams of 3-9* | *“Micro” teams of 1 or 2* |
| *Self-organising* | *Managerial* |
| *Supported cross-functional teams* | *Dubious of cross-functional teams* |
| *High frequency working, purposeful assembly of people to deliver a shared goal* | *Defined process presuming requirements change, delivering at a given time* |
| *T-shaped skillset preferred* | *Specialisation preferred* |
| Leadership style | *Agile evangelist* | *Highly technical* |
| *Defended team from outsiders defining deadlines and estimates* | *Imposed deadlines and estimates on teams* |
| *Transparency* | *Transparency* |
| *Operates at strategic level* | *Operates at operational level* |
| *Did not 'pretend' to have in-depth knowledge of systems or products* | *Presented as an expert in all systems and products* |
| Personal Traits  (in the view of the participants) | *Philosophical, principled, theoretical* | *Analytical, Logical* |
| *Unpredictable* | *Consistent* |
| *Indirect* | *Direct, blunt* |
| *Knew how to engage people* | *Lacked people skills* |
| *Passionate* | *Mechanistic* |
| Self-identified Philosophy | *Meritocracy, Cooperative Model* | *Hierarchical* |

Nelson and Julius had diverse backgrounds, Nelson as an agile coach and Julius as a subject matter expert and developer. Each required a distinct set of skills and expertise. As an agile coach who spent much of his time on the ground with people, Nelson had developed a strong base of soft skills. He was described as articulate and passionate. He intrinsically focused on agile and agile ways of working, forming strong relationships with the scrum master group through direct coaching and mentoring. Conversely, Julius acknowledged that he was lacking emotional intelligence and was described by others as lacking in people skills. However, having built a strong career as a technologist, he was highly respected not only by developers but the wider business.

On the other hand, Nelson lacked credibility with the developers when it came to understanding SoftwareCorps’ products and technology. Although the developers appreciated that Nelson did not pretend to understand their technology, this impacted how they felt Nelson represented them at the C-level. There were concerns that Nelson could not be persuasive when it came to driving technology decisions and priorities with other executives. Conversely, Julius had a strong development background and was highly concerned with the detail of what was being developed, even contributing directly to the codebase. With a strong technical background and a proven record of delivering products to market on time, Julius approached most situations from the perspective of efficiency. He believed in choosing the right tool for the job rather than following a specific method, and as such was not concerned with whether teams followed an agile approach.

*“Nelson was definitely more of an agile process over knowing our technology sort of guy, whereas Julius is the absolute opposite” (Bart, Dev)*

As was described earlier in this section, each CTO interpreted agile differently. These differences in interpretation can be followed through to how each leader approached and, ultimately, implemented what they saw as agile working. Notable differences between the CTO’s were perspectives on team size and organisation (as presented in Table 17).

Another point of difference between the CTO's were their perspectives on self-organising teams. Participants described how the notion of self-organising, empowered teams was a mantra commonly associated with Nelson. Nelson's tag line was ‘those who do the work should decide how it’s done’. He believed, as is stated in the agile manifesto, that the best outcomes come from self-organising teams of empowered individuals. He actively promoted this across the business. However, there were boundaries to self-organisation, even under Nelson. For example, teams acknowledged they would not be able to revert to a traditional staged development approach. There were also cases where Nelson made decisions on behalf of the department, for example consolidating job titles so that all members of the department held the title of either developer, lead developer or scrum master. This decision proved contentious with those who had held titles such as architect, development manager and tester. Julius, on the other hand, felt self-organisation was at best theoretical. He suggested that it could not be successful in reality unless the team members were all very senior. Consequently, when he became CTO, it was evident that the freedom afforded teams under Nelson to self-organise would no longer be given. Julius set clear directives of how teams must organise, for example, into micro-teams, and how the lead developers would manage them.

There were both positive and negative accounts of the decisions and styles of each CTO. Both leaders had supporters and opponents to their approaches, with participants describing instances where both had alienated parts of the department to a greater or lesser extent;

*“To the extent that Nelson seemed to alienate individuals, Julius seems to try and alienate entire teams” (Ted, LD)*

A challenge for Nelson appeared to be the participants viewing some of his actions as contradictory to the principles he spoke widely of following, such as speaking of self-organisation but also making what some saw as dictatorial decisions. For Julius, some appreciated his direct approach and were more understanding of his dictatorial nature, given his level of expertise. Others, however, found they could not accept such a shift in their way of working, having worked in a self-organising environment under Nelson. Regardless of whether the participants agreed with the approach, most noted that it was unpleasant for anyone who found themselves on the wrong side of either CTO;

*“In the same way that Nelson had bad books, if you’re in Julius’s bad books you’re going to get a lot of stick. There are distinct parallels” (Ted, LD)*

Through participants’ accounts of working under both Nelson and Julius, and by their accounts of being CTO, it became clear that different drivers motivated them. Nelson was described as highly principled, focusing both on agile principles and his own philosophies. Julius described himself as being analytical and logic led, driven by a desire to efficiently deliver based on his own experience. It is unsurprising, therefore, that they dramatically differed in how they approached agile working.

In the following sections, the impact each leader had on agile working practices will be explored in more depth.

## Relationships between Leadership and Agile Working Practices

In the previous section, the leadership style and interpretation of agile of two CTO’s were explored. In the following section, an in-depth analysis will be carried out on how these aspects of leadership influenced the working practices associated with two agile principles; Self-organising teams and Reflect and Improve. These principles were selected for in-depth analysis for two reasons. First, they ranked in the top 3 of the 12 agile principles based on the data. Second, these principles were most often used by participants as examples of the distinction between how they worked under Nelson and Julius.

### Relationship between Leadership and Self-Organising Teams

The extent to which individuals and teams were free to self-organise at SoftwareCorps appeared to be influenced by those in leadership positions. Of particular note were the boundaries or limitations these leaders put in place at the individual, team and organisational level. Different leadership styles were seen to have a supportive or constraining effect on self-organisation, based on the freedom and autonomy afforded at each of these levels. This section is structured as follows; First, an overview of what participants understood self-organisation to be is given. Second, individual, team and organisational freedom will be explored. Within each, the enabling and constraining effects of different leadership styles on a team’s ability to self-organise will be presented. A summary of the relationship between leadership and self-organisation will then be given.

#### Participants’ Understanding of Self-organisation

On the subject of self-organisation, and in particular, how self-organisation related to the adoption of agile working practices, there were a variety of opinions from the participants. For example, Frankie (SM) described how, in his view, the agile principle of self-organisation was central to the adoption of agile. He believed it directed the organisation to move away from heavy, hierarchical management, and gave autonomy to those doing the work; *“switching the management on its head”*. Frankie (SM) argued that the principle of self-organisation *“makes agile work”* because it means putting those doing the work in control of it, which he felt made *“so much sense”*. It was his view that solutions to problems would grow *“organically”* out of self-organising teams who shared a *“common goal”.* It was the culmination of all of these facets that he felt contributed towards the success of an agile software development team.

On the other hand, Harper (PO) questioned why self-organisation was specifically an agile concept, suggesting it was merely a popular *“buzzword”* and not particularly agile at all*.* Similarly sceptical, Richie (C-level) questioned the necessity of following agile principles such as self-organisation fully, stating that teams should pick and choose the bits that worked for them;

*"If somebody says 'look actually this gets us closer to [working efficiently] and there are pieces that I can pick up and use like…how you allocate work on a more flexible basis and a shorter time’, all of which can be very good, [but that] doesn’t mean you have to do everything according to agile” (Richie, C-level)*

Several participants noted that much of self-organisation in principle is based on mindset and attitude, which are not easily taught. Although self-organising teams is a principle presented in the agile manifesto: *“The best architectures, requirements, and designs emerge from self-organizing teams”* (Beck et al., 2001b), it is highly conceptual, and there is nothing in the agile manifesto or scrum guide detailing how to accomplish it. Harper (PO) noted how this could be a challenge in an environment accustomed to clearly defined process and methodology;

*"My general experience with developers is that they will take a process very, very seriously. If you say 'Look, this is how it should be done' they'll do it, and they'll often do it to the detriment of common sense…I just think that it's part of that development mentality. It's very rule-based. So, they’ll take processes seriously, but not necessarily because they hugely believe them, just because that’s the process” (Harper, PO)*

It is evident that without a clear understanding of self-organisation in principle, how this is enacted day-to-day will vary between individuals. The following sections explore the manifestation of self-organising teams at SoftwareCorps under Nelson and Julius.

#### Individual Freedom

Individual freedom, in this context, refers to the freedom afforded an individual to carry out their day-to-day work. For instance, the extent to which they can decide how to implement a task, what order to work on their tasks, and the influence they have on choosing what they work on. To differing extents, both Nelson and Julius were seen to give autonomy to individuals to organise their own work, as will be presented next.

##### Individual Freedom Under Nelson

Nelson’s support for individuals to be free to organise their own work was evident in his vocalisation that those who do the work should decide on how it’s done. It was also evident through participants’ descriptions of the high degree of freedom Nelson gave them regarding how they worked. There was, however, an acknowledgement that there were boundaries to this freedom. For example, regarding the strategic decisions of what was to be delivered. Gabriel (DiT) described how, in his experience, individuals had the freedom to choose the type of work they wanted to do. However, this decision had to be taken within the boundaries of what the business needed to be delivered. For example, Gabriel (DiT) was particularly passionate about test automation, and as such, spearheaded this work on his team. He noted, however, that in some cases he chose to prioritise manual testing over-investment in automation if this ensured the project was delivered on time;

*“Where you’ve got a project which…is UI heavy, and the investment required in UI automation is always going to be large, you might be better off just manually testing everything, which is painful but in terms of prioritisation it does kind of make sense sometimes” (Gabriel, DiT)*

Trust, and in particular leaders instilling a sense of trust in their followers, emerged as an enabling leadership characteristic for self-organisation to develop. Paul (SM) reflected that in his experience at other companies, higher management was averse to devolving responsibility to the team. He believed they feared that individuals would *“run riot”,* when in fact, his experience under Nelson demonstrated the opposite. Increased responsibility and a feeling of being trusted empowered individuals to do a better job. Luke (Dev) noted that although it was important for Nelson (CTO) to instil a sense of trust in his followers, it was also crucial for other peers to follow his example and demonstrate this trust. Luke (Dev) described how his product owner trusted him to make priority calls during the sprint within the constraints of the overall sprint deliverables. This mirrored Nelson’s message that those who do the work should decide how it’s done. Luke (Dev) also described a sense of mutual trust between developer and product owner, which allowed him individually the freedom to apply his skills and knowledge to day-to-day decisions. He found this gave him motivation and an increased sense of purpose.

Under Nelson's leadership, individuals were not only empowered to make decisions about their work but trusted to make the right decisions based on their expertise. This trust encouraged them to take greater responsibility and ownership of their work.

##### Individual Freedom Under Julius

Under Julius, individual freedom was more tightly controlled. Julius only gave individuals freedom in circumstances where they had demonstrated success when allowed autonomy, as defined by Julius. From this perspective, participants noted a distinction between empowering individuals to take responsibility and delegating responsibility. Adrian (SM), for example, recalled how Julius would give ownership of a particular area to an individual to allow them to demonstrate that they were capable of the responsibility, but noted that the decision to do this was Julius’. Adrian (SM) reported that this delegation of responsibility and assignment of ownership emphasised individual performance over teamwork and reduced collaboration between individuals;

*“you get…the chance of responsibility, but you don’t really decide on it. It’s not “Okay, let’s break down this problem and own it as a group.” … it was sort of decided by him almost rather than being a bit more sort of done by the team” (Adrian, SM)*

This was also a topic Frankie (SM) touched upon. It was his view that Julius would personally triage all incoming problems, define solutions to them, and then delegate the completion of tasks in which he was less interested. Frankie (SM) described this as negatively impacting collaboration and felt it resulted in disempowering communities who would once have come together to problem-solve.

##### Summary

Although both Nelson and Julius were reported to give autonomy to individuals to organize their own work, their different leadership styles impacted the extent to which individuals felt this was self-organising. Nelson was perceived as empowering those who do the work to decide on how it is done, and Julius as defining the opportunities for individuals to prove themselves responsible.

#### Team Freedom

Team freedom in this context refers to how free a team are to define their working practices and share decision making collaboratively as a team. It was evident that there were marked differences between how much freedom teams had under the leadership of Nelson and Julius, with Nelson appearing as a proponent to self-organising teams and Julius appearing to ‘undo’ self-organisation. This is discussed in the following sections.

##### Team Freedom Under Nelson

During Nelson's tenure as CTO, participants recalled having ownership of their working practices and feeling trusted and empowered to make decisions. It was commonplace for teams to freely implement new ways of working, both in response to identifying issues and as a form of experimentation to drive continuous improvement. This freedom created an environment where teams had *“control of [their] own destiny”* *(Paul, SM)* and as such, took greater responsibility for how they worked. Describing this, Lance (DiT) explained the *“mind shift”* he observed when teams started taking ownership and responsibility for their work. It was his observation that, under Nelson, developers took on more responsibility and began to *“care more”* about the end-to-end delivery of their work, embracing a self-organising mindset;

*“I definitely saw that as soon as developers started taking ownership…started having that responsibility that they definitely cared a lot more about [their work]… I think there is that kind of mind shift change that you have when you start taking ownership of everything [through] to production” (Lance, DiT)*

Harper (PO) supported this perspective, suggesting that self-organisation *“makes the team feel much more responsible for the product as a whole”* and consequently sharpened the focus on delivery. Joseph (PO) described how under Nelson his team had *“really taken [self-organisation] to heart”* and were free to implement new ways of working based on learning from problems they encountered. Frankie (SM) reiterated this sentiment when speaking of a recent project, which he felt illustrated the freedom afforded to him and his team under Nelson’s leadership. He described how they were empowered to own the problem;

*“I really liked that freedom… I loved that because I went and grabbed [the people we needed] and we came together and we did sessions with people throughout the company on how we could improve monitoring for certain products and those people [in the sessions] were the right people to speak to because they were working on the support side for those products; and then we put together a plan…and I don’t think at any one point there was a manager involved. That was a really, really good self-organising example I think.” (Frankie, SM)*

Frankie (SM) reflected that the success of that project was enabled by the team self-organising, *“downing tools”* and coming together to problem-solve.

##### Team Freedom Under Julius

Participants reported the 'undoing' of self-organisation during Julius' tenure. This undoing was in part due to his top-down approach, which participants felt was in direct opposition to self-organisation. Julius himself described how he believed self-organisation was *“an illusion”,* arguing that it could only be achieved if all team members were senior. It was his view that individuals with greater seniority and experience naturally had more influence and dominance over their team members. He believed this would lead to an informal hierarchy even when self-organisation was theoretically in place. Julius’ perspective on this was evident in his actions. Participants, including Sheldon (LD), noted that decisions once made by the lead developer community were now made unilaterally by Julius, without input from the group.

Of particular concern to participants were the changes made to the organisation of their teams soon after Julius became CTO. Julius modelled what he considered to be the most effective team structure on his belief that more than one or two people could not work in the same area efficiently. Accordingly, when promoted to CTO, one of his first actions was to reorganise the department. Julius established a structure of small ‘micro-teams’, which resulted in larger teams being fragmented into groups of one, two or three developers. These micro-teams were given specific pieces of work to do, which were defined by Julius. Quinn (Dev) highlighted the impact these micro-teams had on the team’s ability to self-organise, arguing that with two or three people, Scrum became an overhead and the scope for self-organisation was reduced almost to nothing. Adrian (SM) agreed, recalling how his team spoke of losing self-organisation in the shift away from team-based working and due to Julius’ top-down approach to management.

A lack of freedom to self-organise was also evident in how the teams were expected to estimate their work. Teams were seen to provide estimates which were met with criticism from Julius, who often believed that they were overestimating. This belief resulted in him providing them with *“an appropriate estimate”* (Quinn, Dev).

##### Summary

As has been presented in this section, it was evident that different leadership styles had a significant impact on the level of freedom afforded to teams when it came to their ability to define working practices and make decisions as a group. Nelson was seen to advocate for and actively empower teams to self-organise. Conversely, participants felt that Julius’ preference for very small teams, and his tendency to dictate what and how they worked, rendered self-organisation obsolete. Julius' preferences, in addition to his scepticism of self-organisation, illustrate how his style of leadership impeded self-organisation.

#### Organisational Freedom

Organisational freedom relates to the extent that those outside of the team influence team activities, and how much freedom the team have to influence the wider organisation.

##### Organisational Freedom Under Nelson

There were a number of methods Nelson employed to facilitate the freedom of teams, including organisation-wide vocalisation of his support for self-organisation and establishment of communities of practice (known as CoPs).Despite Nelson’s support, where teams interfaced with the wider organisation there appeared to be organisational boundaries, as will be discussed later in this section.

Self-organisation was reportedly a subject Nelson spoke of frequently, not only to those within his department but widely across the business, to the extent some considered it his *“mantra”.*

*“[Nelson] always used to repeat the people doing the work should decide. It was kind of a mantra of his and something I’ve heard people repeat within teams, so it’s definitely something that sticks in people’s minds” (Adrian, SM)*

Through Nelson’s vocalisation of this mantra, and campaigning across the business to raise the profile of self-organisation, Adrian (SM) recalled that other individuals within the organisation began to use the same language. This enabled the ideas of ‘self-organisation’ and ‘empowering those doing the work decide’ to *“stick in people’s minds”.* Generating support at the organisational level in this way paved the way for teams to drive change and improvement on the ground. These changes were particularly necessary for areas of the business the teams interacted with regularly. For example, several issues were raised by software development teams regarding the centralised recruitment process adopted across the organisation. There were complaints that the process was lengthy and time-consuming, with problems exacerbated by the process being owned by a recruitment team external to the department. In response to this, Nelson encouraged representatives from a number of software development teams to design a proposal to address their issues. This encouragement resulted in teams taking ownership of technical recruitment and the hiring process for members of the software development department, fully supported by Nelson.

In addition to Nelson's vocalisation of his support, he also established communities of practice (CoPs). CoPs were formed of individuals who shared a common role, such as scrum masters, but did not work in the same scrum team. The purpose of these CoPs was to not only to provide an opportunity for knowledge sharing but also to offer a supportive environment for peers in the same discipline to share experiences and develop their skills. Additionally, CoPs provided a forum to take a broader strategic view of decisions made within individual teams. This view, in turn, facilitated the community in making decisions at a departmental level, helping to drive change and continuous improvement across disciplines. This department improvement was an important distinction from the continuous improvement facilitated by retrospectives within scrum teams, as it allowed cross-pollination of ideas between teams and an opportunity to radiate information throughout the department.

Nelson (CTO) was heavily involved in setting up these communities, initially joining them to coach the groups in what a ‘good CoP’ looked like before gradually moving away to allow the group to self-organise. Over time, a number of these communities were established, including the lead developer (LDCoP) and scrum master (SMCoP) groups. The SMCoP would meet up to three times per week to share problems and ideas. Each SMCoP session was focused around a specific objective, for example learning, knowledge sharing, problem-solving or team building. An additional weekly joint community of practice was held between the SMCoP and LDCoP, often fed by outcomes from team retrospectives. This community provided an opportunity for scrum masters and lead developer working across teams to help solve problems *“for the whole function, not just…one team” (Adrian, SM)*.

Self-organisation was observed to prosper during Nelson’s tenure. This was as a consequence of his vocalisation of support at both the executive and team level, the high degree of trust and freedom he afforded teams, and through the establishment of CoPs empowered by Nelson to drive change. Despite this, there were boundaries to the degree of self-organisation individuals and teams had, even under a supportive leader.

##### Boundaries to Organisational Freedom Under Nelson

During Nelson’s tenure as CTO, certain boundaries to self-organisation were imposed. Examples of this were instances where organisation-wide risks existed, such as key person dependencies[[9]](#footnote-9), or where significant organisational changes were occurring, such as restructuring. In one example, Nelson had identified an organisational risk that the lead developers had become siloed in their teams and, as such, had become key person dependencies. When asked to rotate to different scrum teams to facilitate knowledge sharing and reduce key person dependencies on individuals, not all lead developers were supportive. This rotation, however, was deemed necessary by Nelson to safeguard the business. As such, individuals were not permitted to opt-out. This was seen as contradictory to the principle of self-organisation by some. Others were more accepting of such decisions, accepting that there will always be limits to how much freedom teams can have. Especially if they exist within an organisation where the wider business does not buy-in to self-organisation, as Adrian (SM) described;

*"[Nelson] always did talk about "Well, within the function it's you guys…who decide," and maybe that frees him up to go and sort out some of the more executive-level stuff or more management-focused stuff…if [team leads] were truly taking over the whole thing it would have been…probably too much for the rest of the business." (Adrian, SM)*

A lack of buy-in to self-organisation outside of the software development department caused tension and presented boundaries to the scope of self-organisation. This tension was particularly evident with areas of the organisation the software development teams interacted closely with, thus limiting their organisational freedom. For example, some teams felt there was a 'wall' between technology and delivery, and that in order to be more effective at shipping working software they needed a member of the delivery team embedded within their team. Nelson was an advocate of cross-functional teams, where everyone involved in the creation and delivery of the product are embedded within the team, therefore reducing the number of dependencies outside the team. Although there were short term examples of creating cross-functional teams, such as one project where sales, product, development and delivery people were brought together to form a single team for the length of the project (see section 6.5), this wasn’t something the teams or Nelson were able to implement permanently;

*“One thing that did work briefly on our team was having a [support] guy as part of our team so we know when he’s deploying it, we can help him with it, and just making it so he wasn’t on the other side of the office was good. Now he’s a [support] team lead and he doesn’t have any time for us” (Bart, Dev)*

Despite the reported success of embedding members of the delivery department into software development teams, both Bart (Dev) and Laurie (LD) noted that after a short time these individuals were 'taken back' by the delivery function. Executive management within that department ultimately over-ruled the team’s desire to be cross-functional. This over-ruling illustrates the apparent limits to the freedom the software development teams had to influence the wider organisation. It was evident that the level of support from executive management was a significant constraint.

##### Organisational Freedom Under Julius

It was observed that once support at the top for self-organisation was removed, a lack of sponsorship at the executive level opened up an opportunity for management to revert to historical archetypes, cultivating an atmosphere of 'we know best'. This was acutely evident in the events which took place when Julius replaced Nelson as CTO. Sheldon, for example, (LD) described how the team-owned recruitment process was reverted;

*“So, a specific [change] that’s visible… is how the recruitment has been run. It’s gone back central [under Julius]… Now the open roles are only really known to [Julius] and maybe two others. The [lead developers] don’t really know them anymore… [Julius] has declared we will have people that effectively have no experience, the makers [academy], and “you will train them”. That wasn’t our decision.” (Sheldon, LD)*

Frankie (SM) described his frustrations at the shift back to top-down command-and-control under Julius. He felt that having those at the top defining solutions for everyone down the hierarchy had been demonstrably unsuccessful. He described a need for self-organising companies to focus on buy-in from the top, in addition to at the team level, for self-organisation to succeed;

*“It needs to be bought into [by] the people at the top… When you take that away, i.e. for instance when [Nelson] moved and there’s no one at the top there fighting for that, then people come in and think they know best and they want to push that onto everyone else” (Frankie, SM)*

##### Summary

In summary, leadership appeared to have a significant effect on the level of freedom teams had. This effect was seen in terms of their influence on the wider organisation, such as creating cross-functional teams. It was also seen through how heavily those outside of the team influenced team activities. To some extent, teams were empowered under Nelson to drive changes outside of the team environment, often focusing on points of interface between software development teams and other business teams. However, those in positions of leadership in other departments ultimately constrained this freedom. These constraints caused particular tensions where there was a lack of buy-in to agile and self-organisation in departments working closely with the software development teams, such as the delivery team.

#### Summary of Relationship Between Leadership and Self-organising Teams

There were distinct differences in the leadership styles and interpretations of agile between Julius and Nelson. It was evident through participants’ descriptions of how they worked under each of these leaders that their leadership styles heavily influenced the degree to which teams felt free to self-organise.

Self-organisation was reportedly a subject Nelson spoke of frequently and strongly believed in as a way to create successful and productive teams. Participants also recalled how he strongly supported empowering people to do their jobs, and that to do this effectively, he encouraged self-organisation. His openness and support of these practices led others to adopt the same language. Despite this, boundaries to the extent of self-organisation were evident, even under a leader who was openly supportive of it. The majority of participants, however, accepted that there will always be limits to how much freedom teams can have, particularly when they exist within an organisation where the wider business does not buy-in to self-organisation.

Having support at the leadership level was also emphasised as crucial to sustaining self-organisation, with Frankie (SM) observing that self-organisation could not persist without someone at the executive level championing it*.* The need for this support was evident in the events that took place when Julius replaced Nelson and the subsequent perception by some participants that self-organisation was withdrawn. Sheldon (LD) noted that soon after Julius became CTO, dictatorial decisions were made without input from the lead developers. This included the decision to create micro-teams. Adrian (SM) recalled that his team spoke of losing self-organisation under Julius and that there was no longer a focus on team-based working. Julius exemplified a top-down approach to management which aligned with his own perspective; that there will always be a hierarchyin any organisation.

Boundaries within which self-organisation was constrained were observed at three levels; individual freedom to organise one’s own work, team freedom to define working practices, and organisational freedom to influence or be influenced by the wider organisation.

A screenshot of a cell phone

Description automatically generatedFigure 15: Differing levels of self-organisation

As shown in Figure 15, the freedom afforded by leaders at each of these levels, and the boundaries set by those in leadership positions defined the extent to which teams could self-organise. These boundaries were seen to flex depending on those in the CTO role and the situational context. Ultimately, although the perspective of the leader determined the extent to which they granted freedom, it was evident that there were limitations to the freedom even supportive leaders could grant.

Taking a step back from this case, and presenting a more abstract view of the data, Figure 16 (a component part of Figure 12) illustrates how leadership style may be presented as a spectrum, with a more command and control or transactional style at one end, and a more empowering transformational leadership style at the other. As described in this section, a leader may demonstrate a tendency towards one end or the other of this spectrum, but may also be influenced by wider situational factors. In addition, Figure 17 illustrates a spectrum on which a leader’s interpretation of agile sits, with a focus on the behaviours of agile at one end and a more cognitive appreciation for agile principles at the other. These spectrums, along with how they fit together to form a conceptual model of leadership for agile adoption, are discussed in depth in Chapter 8, Section 8.4.

A close up of a logo

Description automatically generatedFigure 16: Leadership Style

A screenshot of a cell phone

Description automatically generatedFigure 17: Leader's Interpretation of Agile

### Relationship between Leadership and Reflect and Improve

In this section, data will be presented to illustrate how, during their tenure, Nelson and Julius influenced the extent to which software development teams reflected and improved. The implications of this influence on the adoption of the agile principle of reflect and improve at SoftwareCorps will also be explored.

#### Participants’ Understanding of Reflect and improve in the context of agile

When asked to define what 'agile' software development meant, many participants described regularly reflecting and continuously improving as an important principle of agile, and a significant factor when building an agile environment. For some, making reflection and improvement part of their day-to-day work was key to their understanding of agile and, therefore, tied closely to their definition of it. Participants emphasised the importance of reflect and improve as a principle in enabling an organisation to experiment, measure, and change course very quickly – all features they associated with agile working, as the following participants described;

*“[Agile] is all about the feedback…it’s the continuous learning cycle” (Sheldon, LD)*

*“Working with agile has to be focusing on small, measurable goals and self-improvement… breaking [work] down into something that you can manage and deliver and then when you’re done delivering that, thinking…how can I improve next time?” (Laurie, LD)*

Other participants also highlighted how the incremental nature of delivering software in an agile way contributed towards the principle of reflect and improve. Incremental delivery shortened the feedback loop and created a regular cadence for reflection. Participants spoke of the nature of agile working and how, by design, it encouraged experimental behaviour. This experimentation was encouraged through short sprints, regular delivery of value, a mechanism for reflection in retrospectives, and a focus on continuous improvement;

*“by doing it small, doing it incremental you learn, adapt fast, you build the right thing that has value at that moment…It’s kind of like the experimentation science idea of you come up with your idea, you try it out, you reach a conclusion and you decide what to do next, but you do it really fast in software.” (Sheldon, LD)*

Although many participants associated the principle of reflect and improve with the practice of team retrospectives, where teams look back on the previous sprint and reflect on what could have gone better, what went well, and how they might improve going forward, the need for a mechanism of learning through reflection and improvement was described at various levels in the organisation. At the individual level, to reflect on one's own performance. At the team level, to reflect on the team’s performance and whether a product realised its predicted value. At the organisational level to reflect on the overall company strategy.

Participants described three factors which influenced the extent to which meaningful reflection and improvement took place. Firstly, contrasting attitudes towards the principle of reflect and improve. Secondly, the willingness of individuals to seek out and identify opportunities to experiment in order to improve and learn. Finally, the level of support within the environment, mitigating any fear of failure associated with experimentation. The following sections explore the manifestation of reflect and improve at SoftwareCorps under Nelson and Julius.

#### Reflect and Improve under Nelson’s leadership

Adrian (SM) recalled that early in SoftwareCorps attempts at agile working, retrospectives wereineffective due to a lack of empowerment within the teams to actually effect change and make decisions. He suggested that improvements in this area were made when Nelson joined the organisation, with a focus on empowering teams and continuous improvement.Adrian (SM) also noted that Nelson (CTO) would often encourage people to try new things, and would be transparent about his own experimentation, in order to stimulate a focus on reflect and improve across the department*.*

Nelson highlighted that one of the first things he assessed at the organisation was its ability to reflect and improve. He described how he did this by observing whether there was a mechanism for learning, such as evidence of teams adopting a PDCA cycle[[10]](#footnote-10) (plan, do, check and act). Under Nelson, teams were given autonomy to design how they worked and were encouraged to improve continuously by challenging the status quo of 'how things are done'. Consequently, there were many examples of teams experimenting with new ways of working, as well as ideas being cross-pollinated between teams. One small example of this was a team who experimented with different estimation techniques in an attempt to reduce what they perceived as an overhead in the time and effort they were spending on estimation. Traditionally, teams used an adapted Fibonacci sequence, where each number is the sum of the preceding two numbers (0, 1, 2, 3, 5, 8, 13, 21), to assign ‘story points’ to a piece of work. A story point is an estimate of the effort required to implement a user story, roughly indicating the level of complexity involved, and allowing a comparison of the relative sizes of pieces of work. The team found that estimating to this level of granularity was time-consuming while also finding many of their stories were similar in size. Consequently, they proposed trialling the use of t-shirt sizing - assigning a size of small, medium, or large to a piece of work. Having found this worked well for them, they shared what they had learned with other teams, some of whom also changed to this method of estimation. This decision was taken exclusively within teams, without the need to get ‘approval’ from Nelson. As such, it is an example demonstrating how teams felt not only free to experiment but were encouraged to do so.

#### Boundaries to Reflect and Improve under Nelson

There were boundaries and limiting factors on the extent to which teams could make changes, despite support from Nelson. Improving and challenging their own working practices as a team was one thing, but as teams matured and improved, they saw opportunities for improvement elsewhere. Examples of this have already been described in previous sections, such as a desire to improve the recruitment process for technical roles ([page 248](#Recruitment)), and to create cross-functional teams ([page 251](#XFTeam)).

In these examples, teams were unable to drive lasting changes when other departments were involved, often due to a lack of support from those in leadership positions. This illustrates the influence leadership had on the team’s ability to not only reflect but to actually drive improvements. Although participants found value in the ceremony of the retrospective as a way to reflect on events, without the freedom to act on these reflections, the same issues would regularly be brought up. A lack of empowerment to tackle these recurring issues resulted in teams feeling their retrospectives were ineffective, and as such individuals no longer raised issues in this forum.

Another facet of reflect and improve was that of experimentation as a mechanism for learning. Once again, leadership played an important role. How leaders responded to teams whose experiments were unsuccessful fed directly into the willingness of teams to experiment. Participants described how, in the early days of agile working, there was a fear that if an experiment was unsuccessful, this would be seen negatively by the leadership team and that this would lead to negative consequences on the team. Laurie (LD) described how, at one time during the early attempts at agile, teams were expected to publish their retrospectives for management and the wider business to see. This expectation understandably constrained what teams were comfortable discussing and how open and honest they could be;

*“We used to send [a]round retrospective notes to everyone, all team leads…and it was kind of like airing your dirty laundry, whereas that’s supposed to be for team self-improvement, but previously people could be worried like “If I put that in a retrospective is someone going to read it?”” (Adrian, SM)*

Rowan (SM) noted that where teams felt supported by their leaders to experiment this drove out the fear of failure, leading to more experimentation and innovation in terms of ways of working and, ultimately, higher productivity.

#### Reflect and improve under Julius’ Leadership

Julius (CTO) held the belief that human nature could be a challenge when it came to the principle of reflect and improve as in his view, it would require *“self-criticism”* to reflect on one’s own decisions and performance. He believed individuals were disinclined to do this unless pressured to do so by their leaders. Julius suggested that it could be perceived as intimidating or embarrassing to discuss issues which had caused problems. He added that it required teams to *“carve out time”* to reflect and improve, something he suggested was also a challenge where there was an increased pressure to deliver. His perspective on this subject was widely known, as was the example he set when carrying out his role as a lead developer. For example, Bart (Dev) worked with Julius as a lead developer and described how his approach had become more argumentative in response to Julius' approach in order to get his ideas heard. He recalled how under Julius he felt more on edge and developed a tendency to be more defensive about his ideas;

*“with [Julius] you have to constantly justify and defend your position…I’ve got better at arguing…I feel more on guard a lot of the time and…when I work with people who haven’t worked with [Julius], then I have more of a tendency to just steamroller over people and argue, which is only because I’m used to presenting my ideas to [Julius] and have to defend them, brutally” (Bart, Dev)*

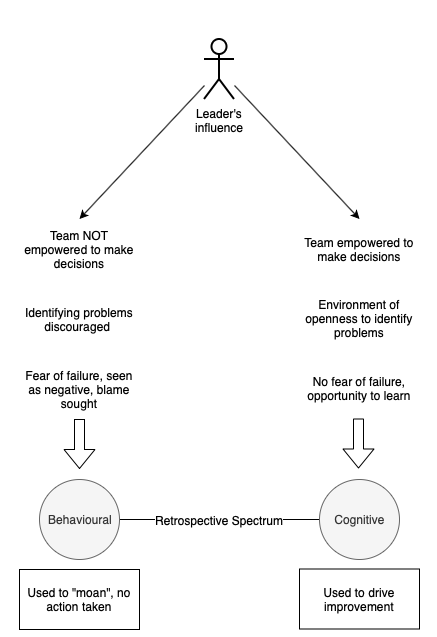
In particular, Bart (Dev) noted that there was a sense of reservation when proposing an idea to Julius. If it did not result in what Julius perceived as an acceptable outcome, the proponent of the idea would be left to deal with recourse in the form of ‘I told you so’. They may subsequently struggle to get support for their ideas or be less likely to suggest them. This was something the researcher experienced directly during an interview with Julius where he gave the following example;

*“So I think my personal growth in this area has been that I know that some errors people need to do themselves before they learn about it. [For example] I found out that I had a scrum master who thought that they could teach people about estimation in [their] own way and at the end sometimes you just need to give people some time before they reconsider their approach” (Julius, CTO)*

In this example, the researcher was the scrum master in question, and as such, the undertone of the example was one of 'I told you it wouldn't work but allowed you to fail so you could learn from your own mistakes'. This exemplifies a distinction between Nelson's leadership style, where individuals were empowered to experiment and were not afraid of failure, and Julius’ style where individuals were *permitted* to try their ideas if they argued for them. In such cases, individuals were largely expected to fail, tying back to Julius’ earlier statement that reflecting on problems could be perceived as intimidating or embarrassing.

#### Summary of the relationship between Leadership and Reflect and Improve

In summary, there was a clear distinction between teams who adopted the behaviours of reflect and improve through carrying out retrospectives, and those who cognitively adopted reflect and improve to drive change and improvement, as illustrated in Figure 18 below.

Figure 18: Leadership and Reflect and Improve

A significant factor influencing this was the role of the leader in creating an environment where teams were empowered to make decisions and act on their retrospectives, in cultivating an environment of openness where problems could be identified without fear of blame, and in supporting experimentation without fear of failure. Teams who did not have this freedom tended to focus only on the aspects of their environment they had influence over. Where teams were repeatedly unable to drive change, this resulted in disengagement with and less willingness to reflect and improve.

Taking a step back from this case, and presenting a more abstract view of the data, Figure 19 and Figure 20 (component parts of Figure 12) illustrate how cognitive and behavioural change can be presented as a spectrum, with behavioural at one end, and cognitive at the other. Both behaviourally enacting and cognitively embracing agile practice and principles is necessary for agile adoption, as described in this section. Figure 19 illustrates the spectrum of desired change, which builds on Figure 16 and Figure 17, and is driven by the leaders interpretation of agile and style of leadership. Figure 20, demonstrates the resulting behaviours within teams, illustrating a cycle of learning through enacting practices behaviourally and growing a cognitive understanding of agile principles. These spectrums, along with how they fit together to form a conceptual model of leadership for agile adoption, are discussed in depth in Chapter 8, Section 8.4.

Figure 19: Change TypeA screenshot of a cell phone

Description automatically generated

Figure 20: Team BehavioursA picture containing screenshot, bird, flower

Description automatically generated

## Summary of Findings

In this chapter, the influence of leadership on the working practices associated with two agile principles (Self-organising Teams and Reflect & Improve) was presented. Table 18 presents a summary of the key points made in this chapter. What follows is a summary of these findings.

Table 18: Summary of Findings Key Points

| **Theme** | **Key Point** | ***Detail*** |
| --- | --- | --- |
| **Leadership** | **Changes in leadership impact on agile working practices** | *Consideration must be given to the impact of leadership changes on in-situ working practices* |
| **Different styles of leadership impact agile working practices** | *Some styles of management are more suited to an agile working environment. Individuals who perceive their role as enabling and empowering rather than managing and dictating are more in-tune with the agile mindset.* |
| **Leader’s interpretation of agile influences agile working practices** | *Where a behavioural view of agile is adopted, focus is on the practices of agile. Where a cognitive view of agile is adopted, focus is on taking an agile approach to problem-solving. This results in differences in how agile is enacted.* |
| **Self-organisation** | **Consider self-organisation as a mindset, not just a behaviour** | *Willingness - Individuals and team members need to understand self-organising principles and be willing to self-organise, for example, promote team solutions over individual ones* |
| *Hierarchy - Willingness of managers/lead developers to enable the team rather than manage them* |
| *People Decide - Empowerment needs to be enabled by the organisation. This requires trust in the team and their outcomes. People who do the work decide. They take “responsibility”. They are focused on delivery.* |
| *Openness - All ideas are valued and evaluated by the team. Nothing is written off without consideration and reason.* |
| *Reward - The way individuals and teams are incentivised must be aligned with self-organising principles to reward the desired behaviours.* |
| *Hierarchy - Specific changes need to be made in organisations to encourage self-organisation. For example, to flatten the hierarchy, to change or remove hierarchical titles. In particular to remove manager, senior, junior etc.* |
| *Share Ideas - Good ideas, solutions, and best practices are shared and cross-pollinated.* |
| **Composition of the team is essential** | *Cross-functional - Teams must be made up of all the skills required to deliver, creating cross-functional teams with end-to-end responsibility.* |
| *Influence - Composition of teams is important. The wrong mix of personalities can result in problems, the dominance of individuals, or over idealised outcomes.* |
| *Coaching - Guidance from experienced agile people, such as coaches, is important. Teams need to have experienced people who understand the principles and how to self-organise effectively.* |
| **Buy-in and support from leadership required** | *Organisational - Clear support and buy-in from the top and the organisation as a whole, not just the IS side, for self-organising principles and enabling of those principles in practice.* |
| *Emergent Leaders – buy-in and support is required from leaders, formal or otherwise, throughout the ranks.* |
| *Boundaries – Self-organisation exits at three levels (individual, team, and organisation). Leaders define the extent of freedom at each level.* |
| **Reflect and Improve** | **Attitudes towards Reflect and Improve** | *Teams must understand the principle of reflect and improve, and its purpose. Degree of contrasting attitudes, depth of understanding of the principle, and engagement in the practice of retrospectives play mediating roles in the extent of meaningful reflection and improvement.* |
| **Willingness to experiment** | *Experimentation is encouraged as a tool to drive improvement.* |
| *Teams must be empowered to address issues directly affecting them, lack of empowerment leads to an inability to experiment and disengagement from reflection and improvement.* |
| **Environment without fear of failure** | *Mistakes are not punished but treated as experimentation. Learning from such situations is critical* |

**Leadership** was found to be a significant factor relating to the adoption of agile working practices, with differences between the style and philosophies of those in leadership positions being critical. In comparing CTO’s Nelson and Julius, it was evident that their approaches differed significantly. Both leaders had supporters and opponents to their style of leadership. Where some participants described Nelson as passionate and transparent, others described him as unpredictable and manipulative. For Julius, there were those who described him as lacking passion and people skills, but others who appreciated his direct and honest approach. It was found that differing perspectives on management, interpretations of agile, and the extent of the leaders’ technical background, all contributed towards changes to working practices described when Julius took over from Nelson in the role of CTO. It was also evident that this change in leadership was a significant influence in the shift away from agile working at SoftwareCorps.

The level of autonomy teams were given to **self-organise** demonstrated how a leader’s interpretation of agile influenced the adoption of associated working practices. Participants’ interpretations of self-organisation varied, from perceiving it as a founding practice of agile working to the belief that self-organisation was purely theoretical. Those who bought-in to the practice of self-organisation noted that it required changes to behaviour and mindset (cognitive and behavioural change), especially when compared to traditional hierarchical management models which tended towards a command-and-control approach. They also noted that change was required not only of individuals within teams but of wider organisational structures such as organisational hierarchy. Teams at SoftwareCorps were seen to have autonomy over their working practices within the software development department under Nelson. However, where they interfaced with other areas of the organisation which were implicitly hierarchical, tensions arose and limited the extent of self-organisation teams could achieve. Teams who were given autonomy to self-organise were reportedly more effective than those who did not, due to their ability to make decisions on the ground. Individuals on self-organising teams reportedly took greater responsibility for outcomes as they were more empowered to do so. The level of control exerted by the individual in the role of CTO directly influenced the extent to which teams were able and willing to self-organise.

Self-organisation was reported to apply at three levels; The freedom of the individual to organise their work, the freedom of the team to define their working practices, and the autonomy of the team to both influence or be influenced by the organisation. Through the comparison of Nelson and Julius’ leadership styles and interpretations of agile, distinct differences were apparent when it came to the level of autonomy teams were given. Nelson, as a proponent of self-organisation, appeared to grant freedom at the individual and team level, although struggled to achieve the same autonomy at the organisational level. Julius, on the other hand, as a sceptic of self-organisation, gave significantly less freedom at the team and individual levels. He was consequently associated with the undoing of self-organisation, thanks to his tendency towards micromanagement and desire to be involved with the day-to-day software development.

Despite a number of differences between these two leaders, there were also instances where their approaches aligned. For example, where the situation called for intervention, Nelson displayed command-and-control tendencies. Where Julius felt proficiency had been demonstrated, he allowed individuals more freedom to make decisions. Leadership, therefore, appears to influence a spectrum on which self-organisation and command-and-control exist at either end, rather than as an either/or state. The situation or context within which this spectrum exists influences the positioning of an individual’s leadership style on this continuum.

Leadership was also seen to influence the extent to which teams adequately reflected and improved. As with the principle of self-organisation, participants were seen to hold differing interpretations of what it meant to reflect and improve. Attitudes ranged from reflect and improve as a core facet of agile working (at all levels of the organisation) to being interpreted merely as the ceremony of a team retrospective. The variation with which individuals interpreted the principle led in turn to differences in how teams operationalised reflection and improvement. Some utilised their retrospectives to drive change, others simply used the ceremony as a chance to air grievances. Where teams sought to drive improvement, experimentation was seen as a mechanism for learning, complimented by the short and focused feedback cycles of agile working.

There were several factors seen to influence the extent to which teams achieved meaningful reflection and improvement. These included whether teams were empowered to drive change, how failure was perceived, and whether there was an environment of openness to discuss issues and problems. These factors were seen to be influenced by the philosophy of, and the example set by, the leader. This, in turn, influenced the extent to which meaningful reflection and improvement took place within teams. Reflect and improve can therefore also be seen as a spectrum, where one end represents teams who adopt only the behaviour of the retrospective, and the other those who demonstrate a cognitive change to use the principle of reflect and improve to learn and drive improvements. Leadership plays a central role in influencing where on this spectrum a team may sit, based on the environment they create and whether they empower teams to make decisions. The next chapter will provide a discussion of these findings in the context of the existing literature, concluding with a model based on findings from this study.

# Discussion

This chapter discusses the main findings of this study and is structured as follows; First, findings contributing to answering each research question are summarised and discussed in relation to the wider research (as detailed in the Literature Reviews in Chapter 2 and Chapter 3). Second, a model of leadership for agile adoption is presented along with the theoretical and practical contributions of this study. Finally, the limitations of this research and recommendations for future research are discussed, along with concluding remarks.

## Introduction

In both the academic and industry literature, agile methods are presented as a solution to decades of ineffective information systems development. However, although agile methods have grown popular since the publication of the Agile Manifesto in 2001, there is significant variability in how they are adopted and used. For example, some organisations adopt agile methods as a fad (Cram & Newell, 2016), and some undergo large scale agile transformations (Korhonen, 2013). The literature has identified a lack of academic research regarding agile practice usage (Dikert et al., 2016), methodology adoption in software engineering (Riemenschneider et al., 2002) and guidance on the adoption of agile (Sidky et al., 2007). There is no widely acknowledged approach or method for adopting 'agile'. Therefore, the puzzle that motivated this study was that if agile software development is so good, why does the adoption of this IS development method vary across organisations?

Given the lack of understanding regarding the variability in agile adoption and the factors which influence it (Strode, 2016), this study aimed to understand how agile working practices and principles may be adopted by software development teams. To explore this, focus was given to *how* agile principles and practices were adopted day-to-day in software development teams, using an in-depth case study approach. Factors influencing adoption in the case under study (SoftwareCorps) were identified, and one critical factor, leadership, was explored in more depth.

The research questions (RQs) for this study were as follows;

**RQ1:** How do the practices of agile software development enable the manifestation of agile principles in software development teams?

**RQ2:** How does leadership style affect the adoption of agile working practices in software development teams?

In the following sections, findings from this study are compared and contrasted with the existing literature. The sections are structured in line with the research questions.

## RQ1: How do the practices of agile software development enable the manifestation of agile principles in software development teams?

To address the first research question, a review of the available literature on agile adoption was carried out, which helped to conceptualise agile principles versus agile practices. Poppendieck and Poppendieck (2003, p. xxiv) defined *principles* as guiding ideas and *practices* as the actions taken to enact those principles. Applying this specifically to an agile context, Campanelli and Parreiras (2015) and Alaa and Fitzgerald (2013) emphasised that agile practices should help accomplish agile principles. With this in mind, agile principles provide the context within which agile practices are carried out. However, after searching the literature for advice on how agile practices facilitate the manifestation of agile principles, a significant gap in the literature was identified.

Studies were scarce regarding how commonly adopted agile practices mapped to the 12 agile principles. The existing literature tended to focus on identifying commonly used agile practices (e.g. Dikert et al., 2016), what practice usage meant in terms of agile maturity (Fontana et al., 2014; Sidky et al., 2007), and on how agile practices mapped to other concepts such as project dependencies (e.g. Strode, 2016) and complex adaptive systems theory (e.g. Alaa & Fitzgerald, 2013). However, to answer the research questions in this study, it was necessary to determine how agile practices mapped to and therefore helped to manifest the agile principles. Given the lack of advice on this in the existing literature, it was also important to address this gap more widely for both academics and practitioners. This was done through a comprehensive mapping exercise, as described in Section 2.3.2 and presented in Table 5.

This mapping exercise contributed firstly to our understanding of the agile principles by demonstrating how day-to-day practices manifest the higher-level principles. This addresses a significant gap in the existing literature not only in terms of how agile principles might be adopted day-to-day but also in building a deeper understanding of the agile principles themselves. Secondly, through applying this mapping to the empirical data from this study, it contributes towards our understanding of how agile practices and principles are adopted in software development teams, as is described below. This was done through two activities; First, through an analysis of how participants perceived the 12 agile principles and the practices associated with them. Second through a detailed analysis of how day-to-day practices mapped to two of the agile principles (self-organising teams and reflect and improve) and the identification of a distinction between behaviourally enacting agile practices and cognitively embracing them. These findings provide a stronger understanding of the practices needed to manifest the agile principles, combining both evidence from the literature and data from the case under study (see Table 5). As such, it guides practitioners in implementing agile practices and informs scholars on how agile principles are adopted.

The empirical findings of this study reveal that through enacting agile practices, software development teams at SoftwareCorps grew their understanding and appreciation of the wider agile principles. Several key findings were identified; First, that agile software development practice adoption was associated with either behavioural or cognitive change by participants. Second, that there were several factors influencing this change which in turn influenced agile practice adoption. Third, that leadership was the most dominant factor that influenced the adoption of agile software development practices in this study. Each of these key findings will be discussed below and compared and contrasted with the literature.

### Behavioural and Cognitive Adoption

As was discussed in Chapter 2, behavioural and cognitive change is an established area of study in the organisational learning (OL) domain. A distinction is made between behavioural and cognitive learning (Leroy & Ramanantsoa, 1997), and in this study these concepts are applied to agile adoption. Cognitive change is concerned with building knowledge, understanding, and insights (Tsang, 1997). Cognitive change is said to redefine values, beliefs, and world views of individuals (Fiol & Lyles, 1985). By contrast, behavioural change is focused on actions (Tsang, 1997) and the adjustment of parameters (Fiol & Lyles, 1985).

Similar concepts to cognitive and behavioural change have been recognised to some extent in the agile literature, although scholars use different terminology. For example, Cram and Newell (2016) describe the *mindful* adoption of agile working practices as a conscious appreciation of complexity along with vigilantly attending to details. By contrast, they refer to the *mindless* adoption of agile working practices as a lack of attention to specifics, where individuals merely follow where they are led. Kostova and Roth (2002) define *ceremonial* adoption of agile working practices as the formal adoption of a practice without the belief there is real value in doing it. Conversely, they describe *genuine* adoption as one where values and beliefs are changed to align more closely with agile practices. As was discussed in Chapter 2, *mindful* and *mindless* adoption (Cram & Newell, 2016) did not appear to give enough focus to the value and beliefs of individuals and, as such, these terms were not used for this study. Although *ceremonial* and *genuine* adoption (Kostova & Roth, 2002) did reflect this perspective to a certain extent, the term ‘genuine’ seemed divisive. Agile software development is based on a set of guiding principles, and to use terms which suggested an end state at which genuine agile is achieved appeared to the researcher to be too simplistic. Additionally, none of the terms used in the agile literature appeared to fit when considering the interaction seen at SoftwareCorps between the enactment of agile practices and the subsequent shift in knowledge, values and beliefs. However, the terms *cognitive* and *behavioural* adoption from the OL literature appeared to inform this perspective and, as such, were adopted for use in this study.

The agile literature suggests that cognitive adoption requires a belief from those adopting new practices that there is value in doing them, resulting in positive attitudes towards the practices being formed (Kostova & Roth, 2002). Additionally, it requires that there is mindfulness, such as reluctance to simplify complex situations, regarding what is being practised and why (Cram & Newell, 2016). Behavioural adoption, on the other hand, is little more than going through the motions. Merely following instructions to carry out practices, often accompanied by strong pressure to adopt and resistance to change (Kostova & Roth, 2002). Ultimately scholars suggest that when the pressure or impetus to adopt is removed, behaviourally adopted practices cease to be used - cognitive adoption is therefore required for change to persist (Cram & Newell, 2016; Kostova & Roth, 2002). Despite identifying the need for cognitive as well as behavioural change, there is a lack of research in the agile domain regarding factors influencing these types of change.

Looking to the OL literature to inform this gap, there is discussion of cognitive and behavioural change as two dimensions of an integrative approach. Behavioural change is seen as complimentary in triggering cognitive change (Leroy & Ramanantsoa, 1997; Zimbardo et al., 1977). It is suggested that by first changing behaviours, such as adopting new practices, it is possible to create new cognitive structures and change beliefs over time (Leroy & Ramanantsoa, 1997). This change is facilitated through modification of organisational routines, and the use of reward and punishment systems, to shape desired behaviours and produce cognitive change (Leroy & Ramanantsoa, 1997). This integrated approach addresses the limitations encountered when cognitive or behavioural change is seen in isolation (Fiol & Lyles, 1985; Leroy & Ramanantsoa, 1997). Where behavioural change alone is seen, this may lead to superficial or short-lived change. Where cognitive change alone is seen, practices may be ineffective when not accompanied by changes in organisational behaviours which become barriers to change (Fiol & Lyles, 1985; Leroy & Ramanantsoa, 1997). Fiol and Lyles’ (1985) model of learning and change (introduced in Chapter 2) presents four positions of cognitive learning and behavioural change, where the change in behaviour can be high or low, cognitive development (learning) can be high or low, and the two positions interact. One position on the model presented by Fiol and Lyles (1985) links closely with the integrative approach discussed by Leroy and Ramanantsoa (1997) – high behavioural change alongside high cognitive development. High levels of behavioural and cognitive change are purported to lead to an increased ability for organisations to deal with turbulent environments and focus on problem solving, as behaviours are changed based on consciously learning what works (Fiol & Lyles, 1985). This highlights the importance of combining changes in behaviours with increased learning and knowledge of how those behaviours are used.

The distinction between behavioural and cognitive change was observed at SoftwareCorps between teams who behaviourally adopted agile practices and those who cognitively adopted agile practices. Teams who behaviourally adopted retrospectives were seen to bring up the same issues repeatedly, however, they did not act to address them. For example, Emmett (LD) spoke of repeatedly raising concerns about the number of features in development concurrently, advocating for focusing on finishing rather than starting new features. He stated, however, that he knew nothing would be done to address the issues and that his concerns would be ignored. Applying what was discussed in the OL literature, this demonstrates a lack of an integrative approach. Participants describing these retrospectives saw them merely as an opportunity to 'moan' or relieve themselves of grievances. This view does not align with the ultimate purpose of the retrospective, which operationalises the agile principle of reflect and improve. Considering this in the context of Fiol and Lyles' (1985) model of learning and change, while behavioural change demonstrated in this example is high, cognitive change is low. As such, actions are not based on learning or knowledge. This has significance to this study as it highlights the limitations of an approach to agile adoption which focuses only on the behaviours and actions of agile practices. Where this is the case, teams can be seen to carry out agile practices, but do not use them to embrace agile principles and ultimately grow their understanding of agile working. They do not make a cognitive change, only behavioural change is observed.

In comparison, teams who cognitively adopted retrospectives were seen to use the practice as an opportunity to drive change, taking actions and designing experiments which were carried into the subsequent sprint. They used an integrative approach, where changes in behaviour were used to learn and create a high degree of cognitive development. This cognitive adoption was also evident through the reactions of teams when retrospectives were delayed. Those teams who cognitively adopted retrospectives reportedly felt a need for them, and as such would object if they were delayed or cancelled. This response was also observed for the daily stand-up and demonstrated a level of cognitive adoption where the team valued and believed in the use of agile practices. One participant (Michael, Dev) asserted that if the existing agile mechanisms were removed, the team would adopt their own *“agile way of doing things”* to ensure they were upholding principles such as face to face communication and self-organisation. Returning once again to Fiol and Lyles’ (1985) model of learning and change, this example demonstrates a high degree of both behavioural and cognitive change, which is most suited to organisations which need to deal with constant change. As such, they prioritise learning and problem solving over stability and predictability. In the example given above, through enacting the practice of the team retrospective behaviourally and subsequently being able to drive change through reflection, a cognitive change occurred, and individuals developed a deeper understanding of the purpose of the retrospective. This led to individuals valuing and believing in the principle of reflect and improve, above and beyond the practice of the team retrospective.

The approach taken to the roll-out of agile practices at SoftwareCorps (as detailed in Chapter 5, Section 5.2.4) initially involved behaviourally enacting agile practices. Teams volunteered to follow Scrum as prescribed in the Scrum Guide (Schwaber & Sutherland, 2017). Following Scrum 'by the book' allowed teams to enact the practices of agile working which, where the environment was conducive, then led to cognitive change – changes to values and beliefs. Changes to values and beliefs were required for the principles of agile to be adopted, rather than merely the behavioural enactment of agile practices.

Applying concepts from the OL literature, this study found that for agile *principles* to be fully adopted, cognitive change is likely to be required rather than merely enacting agile practices behaviourally. This is not a subject which has had significant attention in the agile literature, with a notable lack of focus on behavioural versus cognitive adoption, the need for cognitive change in agile adoption, and factors which influence how cognitive change is achieved in an agile context. There is far more discussion of these concepts in the OL domain, as discussed earlier in this section. Applying these concepts to the agile domain is, therefore, one contribution of this study. Next, the factors found to influence these types of adoption at SoftwareCorps are discussed and compared with the literature.

### What factors influence the adoption of agile working practices in software development teams?

Several factors influential to agile adoption were identified in this study. Table 19 provides a summary of these factors, along with a description of each one and an indication of whether they were evident in the literature reviewed. As shown in Table 19, three of these adoption factors did not appear in the literature reviewed. Therefore, they are presented as adding to the existing body of knowledge. The remaining five factors appeared to some extent in the literature. This section will compare what was found at SoftwareCorps with themes from the literature and highlight key similarities and differences.

Table 19: Adoption factors identified at SoftwareCorps

|  |  |  |
| --- | --- | --- |
| **Adoption Factor at SoftwareCorps** | **Description** | **Appears in Literature** |
| Effective and open communication | Effectively communicating the vision for agile working, and keeping lines of communication open at all levels | X |
| Coaching | Providing a hands-on agile coach as a source of expertise on agile principles, and to coach teams through their issues in real-time. | X |
| Understanding the existing system of working | Taking a considered and informed approach to transitioning to new ways of working | X |
| Removal of barriers | In particular the willingness to remove those barriers seen to inhibit agile working | ✓ |
| Sponsorship and buy-in | At the grassroots as well as management level | ✓ |
| Piloting with volunteers | Which empowers teams to take the first steps in agile working and increases their willingness to tackle issues inherent in learning a new way of working | ✓ |
| Selection of Scrum | Following the chosen method ‘by the book’, enabling a structured approach to enacting agile practices, which leads to an appreciation for agile principles | ✓ |
| Leadership | The impact of changes in leadership on agile working practices | ✓ |

One area where findings in this study align closely with the literature is around sponsorship and buy-in. Factors such as the opinions of users towards the method, the vested interests of employees, the need for management support, and the significant efforts required from both the organisation and teams, could all be considered under the theme of buy-in and support. This is an established and well-researched factor in the adoption literature (Campanelli & Parreiras, 2015; Chau & Tam, 1997; Ichniowski et al., 1995; Ramirez et al., 2007; Riemenschneider et al., 2002) and it is therefore unsurprising to find this was also a factor at SoftwareCorps.

Another theme in the adoption literature was the need for similarities between the new and old practices to ensure practice adoption (Black & Lynch, 2001; Pil & MacDuffie, 1996; Riemenschneider et al., 2002). Discussion in the literature regarding the type of change, ranging from minor evolutionary to revolutionary change (Pil & MacDuffie, 1996), is also considered as part of this theme. For example, ensuring similarities with new and old working practices enables minor evolutionary rather than revolutionary change to occur. The findings of this study challenge the notion that new and old working practices must be aligned to facilitate adoption. The approach taken at SoftwareCorps was to completely remove the old ways of working and take the teams back to basics, focusing entirely on the new way of working and immersing them in the practices of agile. This approach was seen to be successful in facilitating the adoption of agile working practices, in conjunction with allowing teams to volunteer to be trained in the new practices. Volunteers were more engaged and willing to try new practices, and following agile practices ‘by the book’, without attempting to adapt them to existing ways of working, gave focus to understanding not only the practices but the principles of agile working. Although it may be easier for teams to adopt practices which are more familiar to them, agile practices are engineered to facilitate the adoption of agile principles. If organisations change practices to align them with existing ways of working, this may conflict with agile principles. This is especially true where incumbent methods are traditional and undermine the purpose of enacting agile practices in the first place.

Closely tied to the theme of aligning old and new practices, Pil and MacDuffie (1996) identified *“the level of complementary organisational practices and technologies that would increase the benefit from the new practice”* in their study of practice adoption in the auto manufacturing industry. They found that when HR practices complemented the use of high-involvement working practices, usage of those practices increased. Applying this in the context of SoftwareCorps and agile adoption, the level of complementary (or conversely, incompatible) organisational practices was highly relevant. It was found that where teams encountered organisational practices which were incompatible with agile practices, this limited their ability to continue to drive improvements and work in an agile way. In this study, this limitation was described as hitting ‘organisational boundaries’, and the need to address this issue was classified under *removal of barriers* to agile working in the factors identified (see Table 19). An example of these barriers was the need for software development teams to pass releases to the delivery department, which operated under ITIL, before they could go live. This hand-off reduced the ability of the team to release regularly and receive timely feedback from customers, as the ITIL process was tightly controlled and out of their hands. There is evidence, therefore, supporting the literature on the need to ensure wider organisational practices complement rather than conflict with new working practices, to ensure the new practices continue to be used. Where this study adds to the existing body of knowledge is in finding this concept applies to the adoption of agile working practices, not only the adoption of high-involvement working practices in the manufacturing industry.

This study also builds on ideas discussed by Black and Lynch (2001) and Kostova and Roth (2002) that *how* a transition is approached may be more important than the practices selected for adoption. Kostova and Roth (2002) recommended copying approaches seen to be successful elsewhere and adopting patterns which are considered appropriate for the environment, while discouraging coercive adoption imposed by authority. Black and Lynch (2001) emphasised the softer side of approaching adoption, noting, for example, that it was not the practices themselves which increased productivity, but giving employees a voice in decision making that had the biggest impact. Neither of these studies (Black & Lynch, 2001; Kostova & Roth, 2002) were focused on agile adoption specifically, but on practice adoption in general. The findings of this study add to the existing literature by exploring these concepts specifically in terms of *agile* practice adoption.

In the past, SoftwareCorps had attempted to adapt agile practices to fit with their traditional approach, creating 'bespoke' working practices. Changes were imposed on teams from above with the support of external consultancies which were engaged to design the development process on SoftwareCorps' behalf. Participants spoke of how this process had failed to deliver promised improvements to delivery and customer satisfaction and did not have the buy-in or support of the software development teams expected to use it. In comparison, they spoke favourably of adopting agile ‘by the book’, where teams were given the option to opt-in to agile working voluntarily, and where agile principles, such as self-organisation and empowerment, were considered alongside agile ceremonies and practices.

Findings from this study support recommendations by Kostova and Roth (2002) which discourage a coercive approach to adoption. However, these findings challenge the idea that organisations should simply copy what has been successful elsewhere. Findings show that it is not enough to simply enact or mimic agile practices. To sustain agile working in the long term, it is necessary for individuals to make a cognitive shift, to understand the purposes of agile practices and therefore grow their knowledge of agile principles. For teams who understand and embrace agile principles, the adaptation and continuous improvement of their working practices are grounded in learning through reflecting on what works for them. From this foundation, teams may develop their own way of operationalising agile principles rather than merely copying what others have done.

There has also been debate in the literature regarding whether a 'big-bang' approach to agile adoption is more suitable than an incremental approach (e.g. Black and Lynch, 2001; Riemenschneider et al., 2002; Fitzgerald et al., 2006; Fry and Greene, 2007; Solinski and Petersen, 2016). A 'big-bang' approach refers to transitioning the whole organisation to agile working simultaneously, over a relatively short period. Benefits of this approach are suggested to be avoiding organisational dissonance (Fry & Greene, 2007), demonstrating decisive action for adoption (Fry & Greene, 2007), and reducing issues between teams working under different models (Mencke, 2008). Arguments against a ‘big-bang’ approach included not being able to learn iteratively, meaning mistakes or issues would impact all teams at once (Fry & Greene, 2007), constraints on coaches meaning not all teams could be coached day-to-day (Fry & Greene, 2007), management perceiving the risk as too high (Lycett et al., 2003), and lower code and design quality (Solinski & Petersen, 2016).

In contrast, an incremental approach to agile adoption involves piloting agile working with individual teams and iteratively rolling agile out across the organisation, over a longer period. Reported benefits of this approach included positive word of mouth creating interest from the organisation (Benefield, 2008), ‘bootstrapping’ by taking an agile approach to the adoption of agile itself (Wang et al., 2012), increased adaptability in the adoption process (Solinski & Petersen, 2016), and greater facilitation of knowledge and learning (Solinski & Petersen, 2016). Limitations reported are largely the converse of the benefits of a big-bang approach, in particular challenges between teams who needed to work together but are not both following an agile approach (Mencke, 2008).

At SoftwareCorps, an incremental team-by-team approach was taken, allowing dedicated coaching and focus to be given to each team as they transitioned. This approach was seen to be successful in arming the teams with the tools they needed to become self-sufficient, through dedicated, on-the-ground coaching. This incremental approach was a conscious decision for the transition. It included piloting with volunteers, which allowed those championing the agile transition to demonstrate the benefits to those unfamiliar or sceptical of the transition, before a wider roll-out. It also provided the opportunity to address issues and blockers to agile working with early adopters, such as the unavailability of product owners, benefitting teams who came to adopt agile later in the transition. The findings of this study, therefore, support an incremental, team-by-team approach.

As discussed in Chapter 2, the necessity of whether or not to use whole methods[[11]](#footnote-11), rather than selectively implementing practices from different methods, and the importance of the complementarities between practices was a theme in the literature, albeit with conflicting advice (Black & Lynch, 2001; Fitzgerald et al., 2006; Pil & MacDuffie, 1996; Riemenschneider et al., 2002). Arguments for tailoring agile methods, and selectively choosing which practices to adopt and which to ignore, are generally based on the idea that ‘one size’ does not fit all (Abrahamsson et al., 2003; Fitzgerald et al., 2006). A preference for ‘whole’ methods stems from claims by their creators that whole methods are greater than the sum of their parts, and that benefits are achieved through the combination of all the practices – where complementarities are seen (Fitzgerald et al., 2006).

Complementarities between agile principles and their associated practices were observed at SoftwareCorps between reflect and improve and self-organising teams. The principle of reflect and improve encourages teams to reflect at regular intervals in order to identify potential improvements and to encourage experimentation within their working practices. The principle of self-organising teams encourages the creation of autonomous, empowered teams, with authority for decision making. These two principles and the practices which manifest them were seen to interact, and had the potential to create a greater impact when used together; self-organisation appeared to influence the extent to which teams would or could reflect and improve, and the extent to which teams were seen to reflect and improve appeared to influence the freedom afforded to them to self-organise.

For example, where Julius felt teams were not adequately reflecting and improving he saw a need to intervene, either directly or through what he considered to be the senior members of the team (scrum masters and lead developers) who could “enforce” retrospectives on his behalf. An area of particular concern for Julius (CTO) was his perception that teams were not improving the accuracy of their estimates. Participants described cases where teams provided estimates to Julius which he considered *“ridiculous”.* As such, he would intervene by instructing the team in how to organise themselves to deliver based on the estimation he felt was more appropriate. Where Julius intervened, teams became less self-organising as a result of his top-down approach, against which they felt powerless to push back.

Nelson (CTO) was also seen to intervene in this way, for example in the case of a team who had a series of critical incidents (bugs in production which resulted in a loss of service to customers) as they increased their release cadence. The team were perceived not to be making adequate learnings from these incidents, leading to recurrences of critical incidents. As such, Nelson (CTO) deemed it necessary to step in and mandate that the team increase their automated test coverage and document all essential test cases. The team also lost the freedom to release on demand and were forced to hold a release preparation meeting before all releases detailing all changes, tests that had been run, and the expected impact on customers.

The self-organisation of a team, therefore, appears to be influenced by the success of the group at adequately reflecting and improving – a team who are not reflecting on how they work and how they can become more successful leave themselves open for someone else to fill that gap – a more traditional managerial role. It was also evident that when teams were not free to self-organise, they did not take responsibility for owning improvements. Where teams did not have the power to make change happen, this resulted in retrospectives for ceremony rather than purpose as they were unable to resolve issues impacting them day-to-day. The findings of this study suggest, therefore, that although there are benefits from carrying out practices in isolation, there are complementarities found between practices which enhance their effectiveness. As a result, the adoption of whole methods was seen to facilitate the adoption of agile practices and principles, as synergies between these practices increased their effects.

There is limited reference in the literature to the role of leadership as a factor in the adoption of agile working practices (Mishra et al., 2017), the closest being the *“presence of a mandate to use a method”* (Riemenschneider et al., 2002). It is on the subject of leadership and agile adoption that this research adds significantly to the existing body of knowledge, exploring a factor which has had limited attention to date. It was found that changes in leadership led to changes in working practices and that different styles of leadership led to different types of adoption in terms of cognitive and behavioural change. This is discussed in the following Section (8.3).

In summary, the findings of this study concur with the literature that buy-in and support from management (Campanelli & Parreiras, 2015; Riemenschneider et al., 2002) and at the grassroots level (Chau & Tam, 1997; Ichniowski et al., 1995; Ramirez et al., 2007) is essential for the adoption of agile working practices. Additionally, this study supports the literature on the subject of wider organisational practices and the need to ensure these practices complement, rather than obstruct, agile working (Pil & MacDuffie, 1996). However, they challenge the suggestions from some scholars that aligning new ways of working with the old is required (Black & Lynch, 2001), that a big bang approach to agile rollout is preferable (Fry & Greene, 2007), and that practices should be adopted on an as-needed basis, through method tailoring, rather than as whole methodologies (Campanelli & Parreiras, 2015). Rather, this study found that due to complementarities between agile practices, the adoption of whole methods is preferable. In order to provide the coaching and support required to transition to a new way of working, and to enable incremental learning, a team-by-team approach is preferable to a big-bang rollout.

Leadership emerged as a dominant theme in this study, with participants regularly bringing up the subject in reference to the adoption of agile working practices. Leadership was found to be a factor in and of itself, as well as a theme across other factors — for example, sponsorship and buy-in, and removal of barriers. As a result, an additional research question was posed: How does leadership influence the adoption of agile practices and principles in software development teams? This factor was subsequently explored in more depth, as will be discussed in detail in the following section.

## RQ2: How does leadership influence the adoption of agile working practices in software development teams

Through the analysis of data from phase one of this study, leadership emerged as a dominant factor to agile adoption in its own right, with changes to working practices and different iterations of agile working being associated with changes in leadership. Leadership was also seen to be a theme within the other factors identified as influencing agile adoption at SoftwareCorps (see Section 6.14). For example, within the factor of effective and open communication, participants spoke of the importance of those in leadership positions, such as the CTO, effectively and openly communicating their vision for agile working. Therefore, leadership was the focus of phase two of this study, as presented in Chapter 7.

Through an exploration of changes in leadership and the consequences of this on the use of agile practices, two key findings regarding leadership and agile adoption at SoftwareCorps were identified. First, that a leader’s interpretation of agile determined the type of agile adopted in software development teams. Second, that different styles of leadership influenced this adoption differently. In Chapter 7, both the interpretations of agile and leadership styles of two leaders were explored in depth. In this section, these findings will be discussed in relation to the wider literature.

Although leadership is studied extensively, there remains a lack of research regarding leadership in the IS domain, and in particular regarding agile adoption. Gaps in the literature include understanding the suitability of different leadership styles for specific sectors (Ghawe & Brohman, 2016), studying leadership outside of the traditional organisational behaviour domain (Behrendt et al., 2017; Van Knippenberg & Sitkin, 2013), and understanding how leaders influence process (Dinh et al., 2014). Chapter 3 provides a detailed discussion of this.

#### Leader’s Interpretation of Agile

Agile was seen to be interpreted differently by Nelson and Julius, which resulted in differences in how agile was enacted under their leadership. Julius interpreted agile as a tool to be used for project delivery. As such, he focused on the ceremony of agile and was selective about which aspects of an agile approach he supported. For example, he did not believe self-organising teams were necessary for project delivery, however he expected all teams to work in sprints with planning ceremonies. Julius represents a *behavioural* interpretation of agile, where practices and paraphernalia of agile are used, but the principles are not embraced.

In contrast, Nelson was seen to embrace agile practices and principles, taking a deep-thinking interpretation of agile as an approach to problem-solving. This included focusing on empowering individuals, reducing hierarchy and championing self-organisation. Nelson represents a *cognitive* interpretation of agile where behaviours and a deep understanding of agile principles are the focus. This can be distinguished from a behavioural interpretation of agile as it focuses on creating a cognitive shift in teams, rather than simply changing behaviours. Teams who cognitively adopt agile shift from merely following a set of practices to understanding the purpose of those practices and how they enable the manifestation of agile principles day-to-day.

Given the influential role leaders hold, and the power they have to define the constraints within which software development teams work, their interpretations directly influenced agile adoption. It was found that when a leader had a behavioural interpretation of agile, their expectations and support for agile working differed significantly to a leader who had a cognitive view of agile as a way of problem-solving. Therefore, changes in leadership, where leaders held different interpretations of agile, contributed towards changes in how software development teams enacted agile day-to-day.

The wider agile literature identified a lack of clarity and guidance regarding how agile principles should be translated into practice (Dingsøyr & Lassenius, 2016), and how the adoption of agile practices should be approached (Sidky et al., 2007). As such, it is unsurprising to find that there is variation in how agile principles are interpreted and in how agile practices are adopted by practitioners.

There was some (although limited) discussion in the literature on the topic of the interpretation of agile, and difficulties which could arise between those who hold different views. For example, Ghobadi and Mathiassen (2016) focused on how individuals in various roles held different perceptions of knowledge sharing in agile development. They identified problems (which they termed barriers) that existed as a consequence of different perceptions. For example, they suggested that in interpreting ‘welcome changing requirements’, users expected teams to be constantly open to new system requirements, whereas the developers expected requirements to be stable during their sprint (usually 2-week increments). Ghobadi and Mathiassen (2016) focused predominantly on how different perceptions influenced communication and knowledge sharing in agile teams. They did not go as far as determining how this influenced the way agile was carried out in these teams, or how leadership influenced these perceptions. This study demonstrated that a leader's interpretation of agile directly influenced how they implemented agile working, and the level of support they had for the agile principles. This study therefore adds to the scarce literature on the interpretation of agile and suggests how this may create variation in agile adoption. It also addresses a gap in the literature regarding how agile principles are translated into agile practices through the mapping of practices and principles presented in Section 2.3.2.

#### Leadership Style

Closely linked to a leader’s interpretation of agile was their style of leadership. Findings from this study suggested that some styles of leadership may be suited to an agile working environment. For example, individuals who perceive their role as enabling and empowering rather than managing and dictating are more closely aligned with agile principles.

This study applied transformational-transactional leadership theory (Bass, 1985) to the agile adoption context. Through the review of both agile principles and leadership theory, significant overlap was identified between the agile principles and the component parts of transformational leadership; individual consideration, inspirational motivation, idealised influence and intellectual stimulation. Based on descriptions of transformational leadership (Bass, 1997) and agile software development (Beck et al., 2001b) in the literature, it is evident that for both it is essential to provide the support and environment the team need to deliver, to set clear and inspiring goals, to unite people around those goals, to provide coaching, and to encourage the expression of new ideas and challenging of assumptions. For this reason, transformational-transactional leadership theory was selected as the lens through which to view this study. The application of this theory, in this context, enabled a greater understanding of the role of leaders, and in particular leadership style, on the adoption of agile working practices. It also built on established leadership theory to understand the influence of leadership on the adoption of agile.

According to the literature on transformational-transactional leadership theory, a transformational leader empowers and intellectually stimulates individuals to seek a higher order of aspirations (Gumusluoglu & Ilsev, 2009; Li et al., 2012). These aspirations focus on the team and organisational goals rather than purely individual desires (Gumusluoglu & Ilsev, 2009; Li et al., 2012). In contrast, a transactional leader employs a leader-follower dynamic with an emphasis on reward and punishment to elicit desired behaviours (Evans, 1996; Li et al., 2012; Shao et al., 2016, 2012). The literature emphasises that these styles of leadership are not either/or, but rather form different aspects of a full range leadership spectrum (Dansereau et al., 2013; Podsakoff et al., 1990; Shao et al., 2016). Dansereau et al.(2013) suggest that no single approach to leadership is suitable in all managerial situations. Individuals, therefore, may show tendencies towards one style or the other. However, this may shift along the spectrum based on the situation and context. This study applies the concept of a transformational-transactional leadership spectrum (Bass, 1985; Shao et al., 2016) and considers how this relates to variability in agile adoption.

Building on the existing literature, this study suggests that a style of leadership closer to the transformational rather than transactional end of the leadership spectrum may be more likely to compliment agile principles. Thus, transformational leadership may be more likely to have a positive influence on the adoption of agile working practices. The findings of this study also demonstrated that a greater level of agile working and cognitive adoption was carried out under a leader who demonstrated a more transformational leadership style.

By contrast, the findings of this study also suggest that a more transactional leadership style in conjunction with a lack of support for agile may inhibit the ability of teams to use agile working practices, potentially leading to disengagement and the 'undoing' of agile working.

Dubinsky and Hazzan (2010) and Xu and Shen (2016, 2018) both found that leadership style was an important factor relating to success in agile software development, however, neither focused on the leadership style of leaders at the C-level, as was the focus of this study.

Dubinsky and Hazzan (2010) focused on the interactions between a "change leader" (CL) and other parties during an agile transition. They identified the need for a CL to adopt a different style depending on whom they were interacting with, which they refer to as an ad-hoc leadership style. They presented a model of ad-hoc leadership which they suggested could be used to analyse transition and change processes. Despite identifying ad-hoc leadership style as commonly emerging during transitioning to new processes, they did not explore specifically how this model facilitated the adoption of agile working practices. Rather they focused on defining the characteristics, dynamics, and uniqueness of the ad-hoc leadership style, illustrating its use for analysing transition scenarios.

Xu and Shen (2016, 2018) focused on the role of the scrum master and product owner and, as such, were concerned with leadership within the team. Their stated aim was to determine how leadership within teams could provide an appropriate environment and the support needed to help achieve agility. Their findings suggested this was through two learning processes – experiential and vicarious learning. Experiential learning focuses on learning through doing, whereas vicarious learning focuses on learning from experts. Xu and Shen (2018) concentrated on the different roles leaders within an agile team needed to play to facilitate these learning processes; observer, navigator, initiator, cultivator, and external coordinator. However, their 2018 paper presented only preliminary findings, identifying these roles and learning processes, but stopping short of exploring them in depth.

Whilst the studies by Dubinsky and Hazzan (2010) and Xu and Shen (2016, 2018) highlighted the importance of leadership in agile adoption, they did inform how leadership at the management (e.g. CTO) level influences the adoption of agile. This is a central concern of this study, which differentiates it from the existing research and answers calls from scholars (e.g. Karahanna & Watson, 2006) to focus specifically on leaders of the technical function in relation to agile adoption. By carrying out an in-depth, longitudinal study of the effects of different leadership styles of two CTO’s on software development teams, this study builds on existing research summarised here (described in detail in Section 3.3) and applies a different management level perspective.

In order to bring together key insights from this study, a conceptual model was developed, demonstrating how agile may be adopted and the role of leadership in this adoption. This model is discussed in the next section.

## A conceptual model of leadership for agile adoption

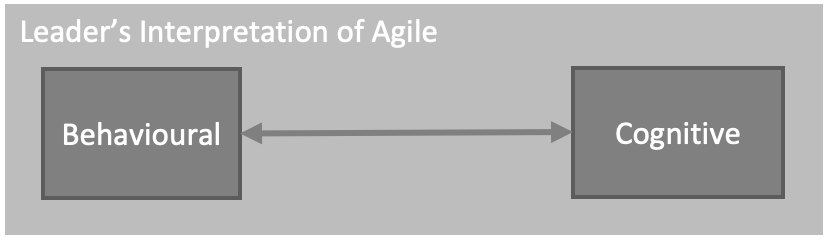
This study's overarching aim was to understand how agile is adopted in software development teams and why there is variability in this adoption between organisations. In this section, an overarching interpretation of how leadership may influence agile adoption is presented based on findings from this study and drawing on the existing literature. A conceptual model visualising these findings is also presented. A discussion of the paths each leader at SoftwareCorps took through this model is then presented. This chapter concludes with suggestions for future research based on this model.

#### Leadership for Agile Adoption

There were several elements considered as influential in the adoption of agile at SoftwareCorps; the leader's interpretation of agile, leadership style, and the type of change leaders’ shaped (cognitive/behavioural). Each of these elements influenced team behaviours and, ultimately, the type of agile adoption. In the following sections, each of these elements is discussed in turn. Each element is presented as a spectrum, for example, ‘interpretation of agile’ ranges from a *behavioural* view of agile to a *cognitive* view of agile. Depending on the position of each element on the spectrum, the outcome and resulting type of agile adoption may differ. Evidence from this study suggests that the leader's influence, in terms of their leadership style and interpretation of agile, may dictate the resulting position of the elements on each spectrum.

#### Leader’s interpretation of Agile

Based on findings from this study, it would appear that how a leader interprets agile heavily influences their actions regarding the use and adoption of agile working practices in software development teams. The spectrum of agile interpretation presented in this study (Figure 21) ranges from a behavioural to a cognitive view of agile.

Figure 21: Leader's Interpretation of Agile

There are a number of tools and methods under the agile umbrella (Alaa & Fitzgerald, 2013). However, a higher-level philosophy, demonstrated by the accompanying principles and values of the Agile Manifesto, also exists. As will be discussed in more detail later in this section, evidence from this study suggests that there is a distinction between merely carrying out agile practices and using agile practices to facilitate the manifestation of agile principles. For example, through carrying out the daily stand-up, the principles of face to face communication and self-organisation can be manifested (see Section 2.3.2 for a review of the literature on agile practices and principles).

Literature on how different interpretations of agile influence agile adoption is scarce, although many scholars note that there is no universally agreed definition of agile software development (Alaa & Fitzgerald, 2013; Conboy & Fitzgerald, 2004). Alaa and Fitzgerald (2013) argue that many agile methods emphasise abstract principles over concrete guidance, focusing on concepts and culture, rather than software practices. Given this lack of definition, it is unsurprising to find variation in how agile practices and principles are interpreted. The adoption of new working practices can be time-consuming and challenging to achieve, and it has been recognised in the literature that transitioning to new ways of working is likely to slow teams down before they are seen to speed up (Dikert et al., 2016).

When agile is interpreted merely as a set of practices to follow to facilitate the delivery of software, there may be less impetus to persevere with practices which are initially difficult to adopt. This interpretation may ultimately influence the type of agile adoption achieved. For example, if agile practices are perceived as tools for software development, there are many alternative methods which could be followed. Other methods may be more easily adopted – for example, those closer to the existing ways of working (see Section 2.3.1 for details of other agile methods). Leaders who perceive agile behaviourally, therefore, may be less likely to encourage teams to persevere where an easier path may be available or to invest the effort required in moving to an agile working environment.

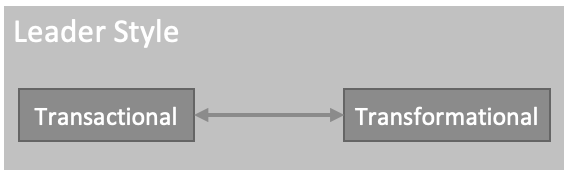
Julius demonstrated a behavioural interpretation when describing agile as a *“process”* and stating that he believed in selecting the “*right tool for the right job”* over adhering to a particular agile philosophy. Examples of this included Julius’ expectation that teams held ceremonies such as stand-up, sprint planning, and retrospectives. In contrast, Julius did not support cross-functional teams or self-organisation, which he believed were unnecessary or impossible to achieve in reality.

When a leader’s interpretation of agile is at the cognitive end of the spectrum, their motivations for adopting agile working practices go further than enacting the practices and paraphernalia of agile. There is a focus on coaching, empowering and facilitating teams in their adoption of agile, not only as a tool but as an approach to problem-solving. As such, the leader’s actions and the expectations they set for their followers reflect this. As a leader’s interpretation of agile moves along the spectrum towards ‘cognitive’ agile, there is also a greater need for leaders to work across the organisation to garner support and educate the business to address potential barriers to agile working. Cognitive agile requires a more systemic view, encompassing the end-to-end product lifecycle from the initial idea through to customer delivery and feedback. There is a greater need, therefore, to address barriers to agile working in the wider organisation and for leadership to support and drive these changes.

Nelson demonstrated a cognitive interpretation of agile through his focus on coaching teams to grow their knowledge and understanding of agile principles, not simply the adoption of a set of practices or a process.

#### Leadership Style

This study draws on transformational-transactional leadership theory and presents leadership style as a spectrum, with transactional at one end and transformational at the other (Figure 22).

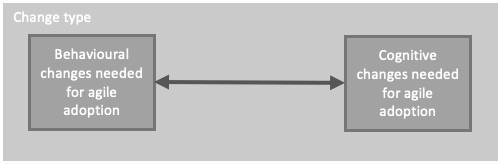
Figure 22: Leadership Style

Leadership theory suggests that these two styles are in fact dimensions of leadership, not opposites (see Section 3.4.1 for more details). Taking this into consideration, the leadership style spectrum presented here is not attempting to show an either/or choice, but rather a range across which individuals demonstrate a general tendency. In addition to their general tendency, leaders may still be seen to demonstrate traits across this spectrum based on context and situation. It could be said that the leftmost end of each spectrum marks the entry point, and as individuals move along the spectrum, they build on these foundations, evolving towards the rightmost end of the spectrum.

Consequently, a leader who demonstrates a transformational style may, in some cases, be seen to take a transactional approach. However, it is less likely that a predominantly transactional leader would adopt a transformational style. Based on where on this spectrum a leader sits, how they influence their followers will differ significantly. As has been discussed already (see 'Agile and Transformational Leadership’, page 96), a transformational leadership style is more aligned with the principles of agile and, as such, will influence the ultimate outcome of agile adoption achieved. Types of agile adoption will be discussed later in this section.

#### Change Type

As was discussed earlier in this chapter, there are both behavioural and cognitive changes required for agile adoption. This is illustrated in Figure 23 below.

Figure 23: Change Type

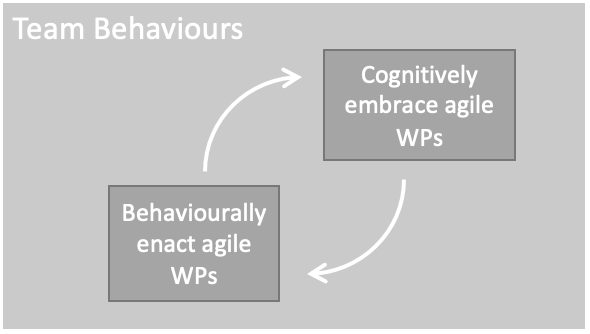
While the practices of agile can be enacted at face value, ultimately, they are designed as the manifestation of agile principles. To embrace these, cognitive change is required such that an individual’s values and beliefs align with the principles of agile. Both a leader’s interpretation of agile and their leadership style influence the type of change. For example, where a leader has a behavioural interpretation of agile, the emphasis is on behaviourally carrying out agile practices. Similarly, where a leader demonstrates a transactional leadership style, focus is given to carrying out mandated behaviours (stand-up, sprint planning, retrospective).

Conversely, where the leader has a deep-thinking interpretation of agile, there is an understanding that cognitive change is necessary in order for followers to embrace the principles of agile (e.g. self-organisation and reflection). A transformational leadership style positively influences this through empowering and inspiring individuals to make this cognitive shift. The outcome of a leaders influence on the type of change observed is manifest through team behaviours which are discussed next.

#### Team Behaviours

The type of change a team has made (behavioural or cognitive) is manifest, and therefore evident, through team behaviours. Drawing from the OL literature, cognitive change requires values, beliefs and world views to be redefined (Fiol & Lyles, 1985) whereas behavioural change is focused on actions (Tsang, 1997) and the adjustment of parameters (Fiol & Lyles, 1985). Building on this, it is to be expected that during early adoption, teams would focus on the basic practices of agile working in order to build their understanding and experience of an agile approach.

In their early adoption of agile working practices, software development teams could be described as taking a behavioural approach to agile adoption. During this time, it is likely that changes to their working practices are most acute and require significant adjustments, as noted by Dikert et al. (2016). As teams experience these practices, however, they move along the spectrum (Figure 24) and grow their understanding of agile as not only a set of practices but a set of principles guiding their approach. Through this process, they internalise agile principles, resulting in a focus on the values and beliefs of agile working rather than merely following a process. For example, a retrospective or daily stand-up can be held in practice, but the value a team achieve through enacting retrospectives is dependent on how they use them.

Figure 24: Team Behaviours

During the early stages of agile adoption at SoftwareCorps one team, in particular, demonstrated the behavioural adoption of the daily stand-up. A senior member of the team had to ask everyone to attend, and team members spoke of attempting to fill their scrum board with post-it notes so that it was impossible to see what was actually in progress. In contrast, teams who cognitively adopted the daily stand-up willingly attended stand-up without being prompted, and their scrum boards were used to clearly radiate information about what was to do, in progress, and done.

As teams exist within the constraints set by their leaders, team behaviours are ultimately influenced by the leader’s interpretation of agile and leadership style. The effects of this on the ultimate outcome of agile adoption will be described in the following sections.

#### Agile adoption outcome

Differences between ‘types’ of agile adoption evident in practice have been discussed in the literature (see 2.4.2). For example, Cram and Newell's (2016) categorisation of agile dabblers, crusaders and tailors, the ‘flavours’ of agile referred to by Fitzgerald et al. (2006), and scholars who refer to the religions of agile (e.g. Williams, 2012). Although these scholars use different terms, they all emphasise the existence of variability in adoption and practice usage. Despite identifying and categorising different types of agile adoption, research is lacking regarding what influences the variability in the types of agile adopted in practice.

Combining each of the figures presented independently in this section, Figure 25 illustrates how each of these elements may relate to each other. It indicates that different combinations of these elements may interact to result in differences in the agile adoption outcome. Table 20 provides illustrative empirical evidence to support the conceptual model, pulling together evidence from the findings chapter and mapping this to each of the building blocks of Figure 25. The model of leadership and agile adoption presented in this chapter (Figure 25) goes some way to explaining the variability in agile adoption, given there are several elements which feed into the ultimate outcome of agile adoption.

A screenshot of a cell phone

Description automatically generatedFigure 25: Conceptual Model of Leadership for Agile Adoption

Table 20: Incidents and Key Evidence

|  |  |  |
| --- | --- | --- |
| **Incidents** | **Key** **Evidence** | |
| **Leaders Interpretation of Agile** | **Behavioural**  *“Agile means we have a process that’s built on the assumption that requirements change over time…at the core agile tries to deliver stuff at a given timescale with a, to some degree, variable scope. But it's not completely open. So, the goal is always to at least follow the Pareto principle* *that you provide 80% of the functionality.” (Julius, CTO)* | **Cognitive**  Nelson described agile as *“high-frequency working and anything else that comes with it”.* He believed agile was the *“purposeful assembly of people to deliver a shared goal”.* His definition of agile and the way he believed it should be implemented focused heavily on the human side of agile, such as empowering those who do the work to define the work, creating a purposeful environment, and building a *“learning company”*. |
| **Leader Style** | **Transactional**  *"When we look at what he's proposing and conclude that it cannot be done, at that point he will come down and get cross and tear apart all of our plans and try to find ways to make it fit his grand design… It's a matter of "This is your job and you will do it and you will meet my expectations," which is not brilliant, to be honest... At my level he's not very interested in my buy-in beyond this is your job and you will do it." (Darren, Dev)*  Julius was seen to monitor an individual's performance actively and take corrective action if deviations occurred. He was also seen to use contingent reward, where individuals were set explicit performance goals for which they would be rewarded. | **Transformational**  *“I have the fundamental belief…that an outcome of a groups work, regardless of how mediocre they are, it is far more superior than the outcome of a single genius because single genius is susceptible to be run over by a bus. The group, however mediocre… their outcome is sustainable. That’s why I genuinely believe in the German school of philosophy in terms of Gestalt, and genuinely because I have a more social democrat leniency” (Nelson, CTO)*  Nelson also noted that, to him, the word ‘leadership’ was misleading. He preferred instead to describe his role as *“facilitator, coordinator, enabler”,* emphasising his ultimate goal to enable individuals to deliver value. |
| **Change Type needed for agile adoption** | **Behavioural**  Julius had a behavioural interpretation of agile, and therefore his emphasis was on behaviourally carrying out agile practices. He demonstrated a transactional leadership style, and consequently focus was given to carrying out mandated behaviours (stand-up, sprint planning, retrospective). This drove behavioural change. | **Cognitive**  Nelson had a deep-thinking interpretation of agile, with an understanding that cognitive change was necessary in order for followers to embrace the principles of agile (e.g. self-organisation and reflection). His transformational leadership style positively influenced this through empowering and inspiring individuals to make a cognitive shift. This drove a cognitive shift. |
| **Team Behaviours** | **Behaviourally enact**  “It was just a case of you were repeating yourself because the things that were an issue for you two weeks ago were still an issue for you. So we hadn't done anything with them, and nobody really wants to sit around and talk about the same problem" (Adrian, SM) | **Cognitively embrace**  *“So retrospectives are kind of where these things get aired…we discuss it and we would almost have a list of actions that we have to take. “If we want to address these particular points how do we address them?” So maybe we need to have a meeting or maybe we figure out what the solution is during the retro and we’d just try that and if that doesn’t work we’ll try something else and we’ll keep trying until we can improve things” (Joseph, PO)* |
| **Agile Adopted** | **In name only**  The language and paraphernalia of agile is used, but without the underlying support for the principles they are intended to manifest:  *“[Julius’] ideas of how to do things are, from my point of view, not scrum. They’re waterfall or they’re manager top-down ideas. It’s basically not self-organising at all… technology isn’t working in an agile [way]...The only thing that’s agile about our technology work at the moment is the fact that we’ve got boards up” (Frankie, SM)* | **Embraced**  Not only are the practices and paraphernalia of agile evident, but more importantly the principles of agile are embraced in the way teams use these practices – a cognitive shift has occurred.  *“If you were to pull agile away and stop doing it, the team would adopt its own agile way of doing it, a variant of it if you will, but they’d find a way to communicate” (Michael, Dev)* |

### Paths through the Conceptual Model of Leadership for Agile Adoption

What follows are two examples of paths taken through the model presented in Figure 25, based on data from SoftwareCorps, which resulted in two different 'flavours' of agile adoption. This is presented to provide clarity of the concepts discussed in the previous sections and to demonstrate how different leaders may create different paths through the conceptual model developed from this study.

#### Nelson’s Journey

Nelson's interpretation of agile was positioned at the cognitive end of the spectrum. Both his own interpretation and how others perceived his interpretation demonstrated that he saw agile as much more than a development methodology. He focused on aspects such as empowering people, the purposeful assembly of people to deliver a shared goal and building a learning company. His vision for agile working also expanded beyond software development and included the wider organisation.

This vision, in turn, influenced the type of change he initiated in the software development department. Through factors such as open communication, coaching, and piloting with volunteers, Nelson demonstrated how an agile way of working was more than merely adopting a methodology. He sought to not only change the behaviours of those in his department but also to facilitate a cognitive shift such that individuals embraced the philosophy of agile working. His position on the 'type of change' spectrum was, therefore, towards the cognitive end of the spectrum.

Nelson's leadership style complimented both his interpretation of agile and the type of change he sought to achieve. He was positioned towards the transformational end of the leadership spectrum. As discussed previously (see Chapter 3 - Agile and Transformational Leadership), transformational leadership and agile principles share many similarities. However, although Nelson predominantly demonstrated characteristics associated with a transformational leadership style, there were examples where he acted in a more transactional way. For example, his decision to rotate the lead developers to different teams, which was taken against the wishes of some of those affected. This decision demonstrates that although leaders may be characterised as having a particular leadership style, it does not mean that they exclusively demonstrate those characteristics. Their style of leadership may alter depending on the context or situation, for example, increased risk or pressure. A leader's actions are influenced by their style of leadership, and this, in turn, shapes team behaviours.

Nelson leadership style, his interpretation of agile, and the type of change he sought to achieve all influenced his actions and, subsequently, team behaviours during his tenure. As was discussed in Chapter 6 (Phase One Findings – Agile Principles), the adoption of agile in teams at SoftwareCorps was assessed through their use of agile working practices. These practices were viewed as the manifestation of agile principles in day-to-day work, as conceptualised in the literature and validated through the phase one findings of this study. Through enacting agile practices, teams achieved a greater understanding of the agile principles. The influence of a leader who supported and encouraged teams to embrace agile working also enhanced the level of cognitive change. Under Nelson, teams were seen to not only enact agile practices but to embrace agile principles. For example, before Nelson's coaching, one team described how their aim was to fill their Scrum board with post-it notes to cover the entire board, rather than to use it as intended to radiate information about their day-to-day work. Later, scrum boards were actively used by all teams as a resource for teams to communicate and visualise their work in a meaningful way.

Through a philosophical interpretation of agile, transformational leadership style and driving cognitive as well as behavioural change, Nelson shaped team behaviours is such a way that they adopted agile principles and values, not merely agile practices and paraphernalia.

#### Julius’ Journey

Julius' interpretation of agile was positioned towards the behavioural end of the spectrum. His approach was to select the 'right tool for the job' rather than adhering to a particular process or methodology. As such, he did not advocate following agile methods in their entirety, but rather for taking a selective approach based on what was most likely to achieve the desired outcome. His view of agile was, therefore, focused on the activities of software development and the general assumption that requirements change over time.

Given his views on agile, Julius created an environment in which it was more difficult for teams to embrace the principles of agile working as they would encounter barriers and a lack of support. For example, an inability to self-organise or drive change through their retrospectives. What resulted, therefore, was a tendency for teams to disengage and simply carry out the behaviours of agile working to be 'seen' to follow the process.

Julius’ leadership style was towards the transactional end of the leadership spectrum. He took an active role in managing both the activities and behaviours of the software development teams, setting clear expectations and defining how they should work based on his own experiences. For example, Julius set expectations regarding how a team should deliver a given project and how many developers were required at a given point in the development lifecycle. Deviations from this were actively corrected and monitored, and reward was contingent on performance as defined by Julius.

Both Julius’ interpretation of agile and his leadership style heavily influenced the behaviours of the software development teams. For example, where teams had once embraced agile practices such as retrospectives, this became difficult to maintain in an environment where they were not empowered to act on their reflections. As a result, teams began to enact practices rather than embrace agile principles, such as using retrospectives to complain without acting to address issues.

What resulted from this combination of variables was the adoption of agile 'in name only'. While Julius maintained that an agile approach to software development (where requirements are assumed to change constantly) was being carried out, he did not subscribe to an agile philosophy. He did not believe that the principles of self-organisation, empowering autonomous teams, or cross-functionality were effective or necessary for the delivery of software. As such practices associated with these principles eroded over time as support for them fell, and barriers to them increased.

### Summary

In summary, there are a number of elements across which leadership may influence the adoption of agile working practices in software development teams; the leader’s interpretation of agile, the style of leadership, and the types of change (behavioural or cognitive). Based on the path taken by a leader through the model above, the outcome of agile adoption may vary, from ‘agile in name’ to ‘agile embraced’. Two leaders were observed, and the resulting impact of their influence on agile working witnessed. Ultimately, it appeared that a deep-thinking interpretation of agile, complemented by a transformational leadership style, may result in a cognitive shift within software development teams, where both the practices and principles of agile are embraced.

In contrast, a behavioural interpretation of agile, along with a transactional leadership style, appeared to result in teams merely enacting agile practices behaviourally and disengaging from the underlying purpose of those behaviours. Evidence suggests this may result in agile being adopted 'in name' only, where the practices and paraphernalia of agile are evident, but the principles are not embraced. In this environment, findings suggest that agile practice usage may erode.

It should be noted that additional paths through this model are possible, for which different outcomes may be seen.

# Conclusion

This chapter summarises the key contributions to theory and practice that have emerged from this study. It is structured as follows; First, the contributions to theory are presented. This is followed by the contributions made to practice. Limitations and recommendations for future research are then provided. This chapter ends with concluding remarks.

## Contributions to Theory

Regarding contributions to theory, this study makes five key contributions;

**First**, this study identifies leadership as a significant influence on the adoption of agile working practices. Transformational-transactional leadership theory is extended through its application in an agile software development context, where leadership research is lacking (Karahanna & Watson, 2006). This study, therefore, demonstrates how leadership style may be conceptualised as a spectrum (as opposed to a dichotomy) with transactional and transformational leadership at opposing ends. This study shows that a leader may manifest a combination of transactional and transformational traits and that categorising a leader as either transactional or transformational may be an oversimplification of their leadership style. In so doing, this study responds to criticism from scholars that research in this domain is typically a-theoretical (Chuang et al., 2014; Cram & Newell, 2016; Dingsøyr et al., 2012; Lang et al., 2013; Strode, 2016). This study also directly contributes towards our understanding of how leadership influences the variability in agile adoption, and suggests that a transformational leadership style is more suitable for agile adoption.

**Second**, a further contribution to theory was made through the presentation of a conceptual model of leadership for agile adoption. This is the first time such a model has been developed. Driven by leadership theory, this model demonstrates how the style of leadership and a leader's interpretation of agile may influence the type of change seen in software development teams. Further to this, the model demonstrates how the type of change, cognitive or behavioural, may drive the type of agile adoption achieved, on a spectrum from ‘Agile in Name’ to ‘Agile Embraced’. This model explains why variation in the adoption of agile working practices exists across organisations, as leaders and leadership style are variable. These findings contribute significantly to existing research on leadership in an agile context, which is currently lacking, by suggesting that some leadership styles are more suited to an agile environment than others. Transformational leadership was found to share many commonalities with agile principles and, as such, is recommended as a suitable leadership style to facilitate agile adoption.

**Third**, through the mapping of commonly used agile working practices to the 12 agile principles, this study provides a richer understanding of the agile principles and how these can be operationalised. Generally speaking, principles are universal but difficult to apply, whereas practices give specific guidance on what to do (Poppendieck & Poppendieck, 2003, p. xxiv). This study addresses a lack of research regarding how agile practices are used to apply agile principles, and on understanding how the agile principles apply in practice. This contribution advances academic knowledge regarding how agile principles are adopted by demonstrating that the enactment of agile practices manifests agile principles in day-to-day work, and illustrates specifically which practices facilitate the operationalisation of each principle.

**Fourth**, within the agile adoption literature, there has been criticism that research lacks theoretical underpinning. This study addresses this concern and contributes towards our understanding of transformational-transactional leadership theory and cognitive-behavioural change through their application in an agile adoption context. These concepts have rarely been used in this domain and scholars have called for more studies focusing on how different styles of leadership may be suited to different contexts (Ghawe & Brohman, 2016).

**Fifth**, this study furthers our understanding of agile adoption. It helps to explain the variability of adoption across organisations by demonstrating that cognitive change is required for agile practices and principles to be adopted as more than a fad. Agile practices were found to be the manifestation of agile principles in day-to-day work, and through enacting agile practices, teams were seen to develop a greater understanding and appreciation for agile principles. Therefore, while agile practices can be adopted behaviourally, it could be said that agile principles are only seen to be embraced when agile practices are cognitively adopted.

## Contributions to Practice

This study offers insight for practitioners regarding how software development teams adopt agile practices and principles and the factors influential to this adoption. This study makes four key contributions to practice;

**First**, this study highlights the importance of leadership style, and a leader’s interpretation of agile, to the successful adoption of agile working practices in software development teams. A transformational leadership style is complementary to agile values and principles and, as such, facilitates adoption. The leadership style of those involved in the transition to agile working must, therefore, be a central concern for practitioners. It was found that when a leader embraced agile as an approach to problems solving, rather than a process to be followed, this was influential in facilitating the type of cognitive change that supports agile adoption. Consequently, advice for practitioners is to ensure that the right kind of leader supports the team and that this is a conscious consideration in the transition to agile working.

**Second**, it provides a practical analysis of commonly used agile working practices and how these map to, and subsequently operationalise agile principles. This analysis facilitates the application of agile principles in a real-world context, giving a frame of reference for practitioners seeking to apply high-level agile principles day-to-day.

**Third**, detailed examples of the distinction between cognitive and behavioural adoption are given, providing insight into the variation seen in agile adoption across organisations and teams. This contribution provides insight regarding the complexities involved in agile adoption and offers examples of how both behavioural and cognitive change can be harnessed during the adoption process. It also demonstrates how cognitive adoption is likely to be required for agile adoption to be sustained long term.

**Fourth**, it provides recommendations regarding how the adoption process itself should be approached. For example, taking an incremental, team by team approach driven by volunteers, piloting with early adopters, creating an environment of collaboration, and empowering individuals are supported by the findings of this study. These findings are offered as advice to practitioners regarding considerations for the design of their agile adoption. Practitioners are also encouraged to consider how wider organisational practices, such as reward and incentivisation, complement or inhibit agile working. Findings from this study recommend that organisational practices as a whole must be considered in the context of agile principles to combat potential barriers to agile working.

## Limitations and Recommendations for Future Research

While this study makes some significant contributions, as described in the previous sections, it is important to acknowledge some limitations of the investigation. The following section discusses four key limitations and six potential avenues for future research.

**First**, this study was limited to a single organisation, and **second**, the analysis focused on two leaders at this organisation over three and a half years. The limitation of a single case and two leaders was intentional in order to bound the research and allow for an in-depth and longitudinal perspective. This type of qualitative study provides a large quantity of data, and as such, expanding data collection to more cases or additional leaders was not feasible within the constraints of a single PhD thesis. A central concern regarding the use of a single case as a unit of analysis is the limitation of generalisation or external validity. A single case is not representative of all cases (and is not purported to be so here), and as such theories developed based on them may not prove explanatory in a different context. Despite these limitations, there are examples influential studies which have employed this method, including Markus (1989), which is widely cited in information systems research (Myers, 1997). Where a single case cannot provide generalisability, it can provide particularisation. This particularisation enables the in-depth exploration and observation of phenomena which is ultimately necessary for theory building.

**Third**, while efforts were made to interview a cross-section of people and levels of hierarchy, such as selecting at least two individuals from each role, it should be noted that participation in this study was voluntary. As such, all possible views and perspectives may not have been represented in the data. For example, individuals who were particularly against agile working may not have agreed to be interviewed. Despite this, almost all those approached for interview agreed to participate. Those who did not gave reasons such as pressing project commitments. Importantly, the researcher identified a point of saturation in the data available during analysis, where no new themes were emerging.

**Fourth**, although there are 12 agile principles in the Agile Manifesto, an in-depth analysis was only carried out on the two most frequently mentioned in the data; reflect and improve and self-organising teams. This limitation was intentional to bound the study and was a necessity in order to meet the constraints of a PhD thesis.

In light of these limitations, suggestions for future research include;

* This study identified the potential of complementarities between agile principles and practices, which could reinforce adoption. This was predominantly explored through the in-depth analysis of the principles of self-organising teams and reflect and improve and their associated working practices. Future research should explore all 12 agile principles to determine whether complementarities observed between reflect and improve and self-organising teams also exist between other principles. While findings from this study suggest this would be the case, this cannot be said with certainty. This could feed into to the debate regarding the necessity of adopting ‘whole’ methods versus tailoring methods to the organisation, where the existence of complementarities was questioned.
* Future research should therefore explore whether ‘whole’ methods provide complementarities between practices, which may increase the effectiveness of using them and whether tailoring of agile methods may, therefore, inhibit these complementarities. For example, if some practices are omitted or their essential nature changed. Additionally, research should consider whether method tailoring may contribute to the variability of agile adoption and usage across organisations.
* The possibility of other paths through the ‘leadership and agile adoption model’ presented here, in different organisations and with different leaders.
* How different combinations of leadership style, interpretation of agile, and cognitive or behavioural changes may influence the type of agile adopted could be investigated. For example, what might the outcome on agile adoption be where a leader is positioned centrally for each variable?
* Explore whether leaders who follow similar paths to those seen in this case result in the same or different outcomes in other contexts, and what factors influence any differences observed.
* Further research exploring psychological profiles of leadership which would develop the ‘leaders interpretation of agile’ aspect of the conceptual model in this study.
* Further development of the mapping of agile practices to agile principles presented in this study could also be carried out. A common activity in the literature was the categorisation of agile practices into groupings, such as social and technical. By combining the mappings presented in this study with categorisations in the literature, it would be interesting to determine whether some categories contribute more or less than others in the adoption of agile practices and principles. For example, are practices in the social category more important than those in the technical category?
* A more in-depth exploration of the cognitive/behavioural action cycle in the adoption of agile practices and principles. In particular, it would be interesting to determine whether behavioural change comes first, before a cognitive shift occurs, or whether a cognitive shift is possible initially.

While it is necessary to recognise the limitations inherent in any study, it is also important to recognise the balance between perceived weaknesses and strengths. Where this study cannot provide a representative view of agile adoption in software development, it does provide insight and an in-depth perspective of agile adoption in the real world — something which to date has been lacking in the agile adoption literature.

## Concluding Remarks

The contributions of this study can be summarised as follows;

* A greater understanding of factors influential to the adoption of agile practices and principles in software development teams
* A practical analysis of popular agile working practices and how these map to and operationalise agile principles
* The development of a model demonstrating factors which contribute towards the variation seen in agile adoption in practice
* The combination of behavioural and cognitive change research and transformational-transactional leadership theory, and the application of this in an agile software development context
* The further development of transformational-transactional leadership theory, conceptualising leadership style as a spectrum

This study aimed to determine how agile practices and principles are adopted in software development teams, and understand why there is variability in how agile is adopted and practised. Findings from this study suggest that a key reason agile adoption varies is that leadership changes can result in an increased or decreased focus on agile working. This focus depends on the leader's interpretation of agile and their style of leadership. The findings of this study suggest that transitioning to an agile way of working is more likely to be successful if the right type of leader supports teams in their transition. This success may be dependent on leaders having a style of leadership that compliments agile values and principles (such as empowering, communicating, collaborating, and so on). It may also be dependent on leaders understanding that agile is not merely a tool for project management and software delivery (a behavioural view), but a way of approaching problem-solving (a cognitive view of agile).

Leaders, therefore, appear to be critical to the adoption of agile practices and principles. Where the desire is for organisations to embrace the adoption of agile working in order to achieve the advertised benefits, leaders must be careful in their CTO appointment decisions. Appointing the right person at this level is critical for achieving the desired type of agile adoption.

# References

Abbas, N., Gravell, A. M., & Wills, G. B. (2008). Historical roots of agile methods: Where did “Agile thinking” come from? In P Abrahamsson, R. Baskerville, K. Conboy, B. Fitzgerald, L. Morgan, & X. Wang (Eds.), *Agile Processes in Software Engineering and Extreme Programming. XP 2008. Lecture Notes in Business Information Processing* (Vol. 9, pp. 94–103). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-68255-4\_10

Abbas, N., Gravell, A. M., & Wills, G. B. (2010). Using factor analysis to generate clusters of agile practices (a guide for agile process improvement). *Proceedings - 2010 Agile Conference, AGILE 2010*, 11–20. https://doi.org/10.1109/AGILE.2010.15

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile software development methods review and analysis*. *VTT Publications*. https://doi.org/10.1076/csed.12.3.167.8613

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New directions on agile methods: a comparative analysis. In *25th International Conference on Software Engineering* (pp. 244–254). Portland, Oregon: Ieee. https://doi.org/10.1109/ICSE.2003.1201204

Alaa, G., & Fitzgerald, G. (2013). Re-conceptualizing agile information systems development using complex adaptive systems theory. *Emergence: Complexity and Organization*, *15*(3), 1–23.

Alon, S., & Tienda, M. (2007). Diversity, Opportunity, and the Shifting Meritcocracy in Higher Eucation. *American Sociological Review*, *72*(August), 487–511. https://doi.org/doi:10.1177/000312240707200401

Anderson, D. J. (2004). *Agile Management for Software Engineering*. New Jersey: Prentice Hall.

Angell, I. O., & Straub, B. H. (1993). Though this be madness, yet there is method in’t. *The Journal of Strategic Information Systems*, *2*(1), 5–14.

Antonakis, J., & House, R. J. (2014). Instrumental leadership: Measurement and extension of transformational-transactional leadership theory. *Leadership Quarterly*, *25*, 746–771. https://doi.org/10.1016/j.leaqua.2014.04.005

Avgerou, C., & Cornford, T. (1993). A review of the methodologies movement. *Journal of Information Technology*, *8*(4), 277–286. https://doi.org/10.1057/jit.1993.38

Avison, D., & Fitzgerald, G. (2003). Where now for Development Methodologies? *Communications of the ACM*, *46*(1), 79–82.

Avison, D., & Fitzgerald, G. (2006a). *Information Systems Development: Methodologies, Techniques and Tools*. *Information Systems Series* (4th ed.). Maidenhead: McGraw-Hill Education.

Avison, D., & Fitzgerald, G. (2006b). Methodologies for developing information systems: a historical perspective. In D Avison, S. Elliot, J. Krogstie, & J. Pries-Heje (Eds.), *The Past and Future of Information Systems: 1976–2006 and Beyond. IFIP WCC TC8 2006. IFIP International Federation for Information Processing* (Vol. 214, pp. 1–12). Boston, MA.

Avolio, B. J., & Bass, B. M. (2004). *Multifactor Leadership Questionnaire 3rd Edition*.

Avolio, B. J., Waldman, D. A., & Yammarino, F. J. (1991). Leading in the 1990’s: The four I ’s of transformational leadership. *Journal of European Industrial Training*, *15*(4), 9–16. https://doi.org/http://dx.doi.org/10.1108/03090599110143366

Balthazard, P. A., Waldman, D. A., & Warren, J. E. (2009). Predictors of the emergence of transformational leadership in virtual decision teams. *Leadership Quarterly*, *20*, 651–663. https://doi.org/10.1016/j.leaqua.2009.06.008

Barling, J. (2014). *The Science of Leadership: Lessons from research on organizational leaders*. Oxford: Oxford Universiry Press.

Baskerville, R. (2006). Artful planning. *European Journal of Information Systems*, *15*(2), 113–115. https://doi.org/10.1057/palgrave.ejis.3000622

Baskerville, R., & Wood-Harper, A. T. (1996). A critical perspective on action research as a method for information systems research. *Enacting Research Methods in Information Systems: Volume 2*, (11), 235–246. https://doi.org/10.1007/978-3-319-29269-4\_7

Bass, B. M. (1985). *Leadership and Performance Beyond Expectations*. New York, London: The Free Press.

Bass, B. M. (1997). Does the Transactional-Transformational Leadership Paradigm Transcend Organizational and National Boundaries? *American Psychologist*, *52*(2), 130–139.

Bass, B. M., & Avolio, B. J. (1990). Developing Transformational Leadership: 1992 and Beyond. *Journal of European Industrial Training*, *14*(5), 21–27.

Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001a). Principles behind the Agile Manifesto.

Beck, K., Beedle, M., Bennekum, A. Van, Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001b). Manifesto for Agile Software Development. *Agile Manifesto*.

Behrendt, P., Matz, S., & Göritz, A. S. (2017). An integrative model of leadership behavior. *Leadership Quarterly*, *28*(1), 229–244. https://doi.org/10.1016/j.leaqua.2016.08.002

Benefield, G. (2008). Rolling out Agile in a large enterprise. In *41st Hawaii International Conference on System Sciences* (pp. 1–10). https://doi.org/10.1109/HICSS.2008.382

Beynon-Davies, P., Mackay, H., & Tudhope, D. (2000). A case study of a rapid application development project. *Information Systems Journal*, *10*, 195–216.

Binns, A., Harreld, J. B., O’Reilly III, C. A., & Tushman, M. L. (2014). The Art of Strategic Renewal: What does it take to transform an organization before a crisis hits ? *Sloan Management Review*, *55*(2), 21–23.

Black, S. E., & Lynch, L. M. (2001). How to Compete: The Impact of Workplace Practices and Information Technology on Productivity. *Review of Economics and Statistics*, *83*(3), 434–445. https://doi.org/10.1162/00346530152480081

Blom, M., & Alvesson, M. (2015). All-inclusive and all good: The hegemonic ambiguity of leadership. *Scandinavian Journal of Management*, *31*(4), 480–492. https://doi.org/10.1016/j.scaman.2015.08.001

Boehm, B. (2002). Get Ready for Agile Methods, with Care. *Computer*, *35*(1), 64–69.

Brynjolfsson, E., Hu, Y. (Jeffrey), & Simester, D. (2011). Goodbye Pareto Principle, Hello Long Tail: The Effect of Search Costs on the Concentration of Product Sales. *Management Science*, *57*(8), 1373–1386. https://doi.org/10.1287/mnsc.1110.1371

Buglione, L. (2011). Light Maturity Models (LMM): an agile application. In *12th International Conference on Product Focused Software Development and Process Improvement* (pp. 57–61). https://doi.org/http://dx.doi.org/10.1145/2181101.2181115

Burns, J. M. (1978). *Leadership*. New York: Harper & Row.

Campanelli, A. S., & Parreiras, F. S. (2015). Agile methods tailoring - A systematic literature review. *Journal of Systems & Software*, *110*, 85–100. https://doi.org/10.1016/j.jss.2015.08.035

Cawthon, D. W. (1996). Leadership: The great man theory revisited. *Business Horizons*, *39*(3), 1–4. https://doi.org/10.1016/S0007-6813(96)90001-4

Chakravarty, A., Grewal, R., & Sambamurthy, V. (2013). Information technology competencies, organizational agility, and firm performance: Enabling and facilitating roles. *Information Systems Research*, *24*(4), 976–997. https://doi.org/10.1287/isre.2013.0500

Charlier, S. D., Stewart, G. L., Greco, L. M., & Reeves, C. J. (2016). Emergent leadership in virtual teams: A multilevel investigation of individual communication and team dispersion antecedents. *Leadership Quarterly*, *27*(5), 745–764. https://doi.org/10.1016/j.leaqua.2016.05.002

Chau, P. Y. K., & Tam, K. Y. (1997). Factors Affecting the Adoption of Open Systems : An Exploratory Study. *MIS Quarterly*, *21*(1), 1–24.

Checkland, P., & Holwell, S. (1998). Action Research: Its Nature and Validity. *Systemic Practice and Action Research*, *11*(1), 9–21. https://doi.org/10.1023/A:1022908820784

Cho, J., Park, I., & Michel, J. W. (2011). How does leadership affect information systems success? The role of transformational leadership. *Information and Management*, *48*, 270–277. https://doi.org/10.1016/j.im.2011.07.003

Cho, J. (2008). Issues and Challenges of Agile Software Development With Scrum. *Issues in Information Systems*, *9*(2), 188–195.

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, *81*, 961–971. https://doi.org/10.1016/j.jss.2007.08.020

Chuang, S.-W., Luor, T., & Lu, H.-P. (2014). Assessment of institutions, scholars, and contributions on agile software development (2001-2012). *Journal of Systems and Software*, *93*, 84–101. https://doi.org/10.1016/j.jss.2014.03.006

Cockburn, A. (2002). Agile Software Development Joins the" Would-Be" Crowd. *Cutter IT Journal*, (January), 6–12.

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, *20*(3), 329–354. https://doi.org/10.1287/isre.1090.0236

Conboy, K., & Fitzgerald, B. (2004). Toward a Conceptual Framework of Agile Methods: A Study of Agility in Different Disciplines. In *ACM workshop on Interdisciplinary Software Engineering Research* (pp. 37–44). Newport Beach, CA. https://doi.org/10.1145/1029997.1030005

Conboy, K., Fitzgerald, G., & Mathiassen, L. (2012). Qualitative methods research in information systems: Motivations, themes, and contributions. *European Journal of Information Systems*, *21*(2), 113–118. https://doi.org/10.1057/ejis.2011.57

Cram, W. A., & Newell, S. (2016). Mindful revolution or mindless trend? Examining agile development as a management fashion. *European Journal of Information Systems*, *25*(2), 154–169. https://doi.org/10.1057/ejis.2015.13

Creswell, J. W. (2003). *Research Design: Qualitative, Quantitative and Mixed Methods Approaches* (2nd ed.). Thousand Oaks, CA: Sage Publications.

Creswell, J. W. (2007). *Qualitative Inquiry and Research Design. Choosing Among Five Approaches* (2nd ed.). Thousand Oaks, CA: Sage Publications.

Dansereau, F., Seitz, S. R., Chiu, C.-Y., Shaughnessy, B., & Yammarino, F. J. (2013). What makes leadership, leadership? Using self-expansion theory to integrate traditional and contemporary approaches. *Leadership Quarterly*, *24*, 798–821. https://doi.org/10.1016/j.leaqua.2013.10.008

Demirkan, H., & Spohrer, J. (2015). T-Shaped Innovators: Identifying the Right Talent to Support Service Innovation. *Research-Technology Management*, *58*(5), 12–15. https://doi.org/10.5437/08956308X5805007

Diebold, P., & Dahlem, M. (2014). Agile Practices in Practice - A Mapping Study. In *18th International Conference on Evaluation and Assessment in Software Engineering, EASE’14* (pp. 1–10). London. https://doi.org/10.1145/2601248.2601254

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and Success Factors for Large-Scale Agile Transformations: A Systematic Literature Review. *Journal of Systems and Software*, *119*, 87–108.

Dingsøyr, T., Fægri, T. E., Dybå, T., Haugset, B., & Lindsjørn, Y. (2016). Team Performance in Software Development: Research versus Current Advice. *IEEE Software*, *33*(4), 1–21. https://doi.org/10.1109/MS.2016.100

Dingsøyr, T., & Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology*, *77*, 56–60.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, *85*(6), 1213–1221. https://doi.org/10.1016/j.jss.2012.02.033

Dinh, J. E., Lord, R. G., Gardner, W. L., Meuser, J. D., Liden, R. C., & Hu, J. (2014). Leadership theory and research in the new millennium : Current theoretical trends and changing perspectives. *Leadership Quarterly*, *25*, 36–62. https://doi.org/10.1016/j.leaqua.2013.11.005

Dubinsky, Y., & Hazzan, O. (2010). Ad-Hoc Leadership in Agile Software Development Environments. In *ICSE Workshop on Cooperative and Human Aspects of Software Engineering* (pp. 32–38). Cape Town, South Africa: ACM. https://doi.org/10.1145/1833310.1833316

Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile modeling, agile software development, and extreme programming: The state of research. *Journal of Database Management*, *16*(4), 88. https://doi.org/10.4018/jdm.2005100105

Eseryel, U. Y., & Eseryel, D. (2013). Action-embedded transformational leadership in self-managing global information systems development teams. *Journal of Strategic Information Systems*, *22*, 103–120. https://doi.org/10.1016/j.jsis.2013.02.001

Evans, M. G. (1996). R.J. House’s “A path-goal theory of leader effectiveness.” *Leadership Quarterly*, *7*(3), 305–309. https://doi.org/10.1016/S1048-9843(96)90021-1

Evans, R. (2007). Authentic Leadership. In *The Jossey-Bass Reader on Educational Leadership* (2nd ed.). Jossey-Bass.

Ewen, C., Wihler, A., Blickle, G., Oerder, K., Ellen III, B. P., Douglas, C., & Ferris, G. R. (2013). Further specification of the leader political skill-leadership effectiveness relationships: Transformational and transactional leader behavior as mediators. *Leadership Quarterly*, *24*, 516–533. https://doi.org/10.1016/j.leaqua.2013.03.006

Ewenstein, B., Smith, W., & Sologar, A. (2015). *Changing change management*. *McKinsey Digital* (Vol. July).

Faraj, S., & Sambamurthy, V. (2006). Leadership of Information Systems Development Projects. *IEEE Transactions on Engineering Management*, *53*(2), 238–249. https://doi.org/10.1109/TEM.2006.872245

Fiol, M. C., & Lyles, M. A. (1985). Organizational Learning. *Academy of Management Review*, *10*(4), 803–813.

Fitzgerald, B. (1996). Formalized systems development methodologies: A critical perspective. *Information Systems Journal*, *6*(1), 3–23. https://doi.org/10.1111/j.1365-2575.1996.tb00002.x

Fitzgerald, B. (1998). An empirical investigation into the adoption of systems development methodologies. *Information and Management*, *34*(6), 317–328. https://doi.org/10.1016/S0378-7206(98)00072-X

Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems*, *15*, 200–213. https://doi.org/10.1057/palgrave.ejis.3000605

Fontana, R. M., Fontana, I. M., Da Rosa Garbuio, P. A., Reinehr, S., & Malucelli, A. (2014). Processes versus people: How should agile software development maturity be defined? *Journal of Systems and Software*, *97*, 140–155. https://doi.org/10.1016/j.jss.2014.07.030

Francis, J. J., Johnston, M., Robertson, C., Glidewell, L., Entwistle, V., Eccles, M. P., & Grimshaw, J. M. (2010). What is an adequate sample size? Operationalising data saturation for theory-based interview studies. *Psychology & Health*, *25*(10), 1229–1245. https://doi.org/10.1080/08870440903194015

Friedman, H. H. (2016). Is Your Organization Run by the Right Kind of Leader? An Overview of the Different Leadership Styles. https://doi.org/10.2139/ssrn.2844519

Fry, C., & Greene, S. (2007). Large scale agile transformation in an on-demand world. In *Agile Conference* (pp. 136–142). https://doi.org/10.1109/AGILE.2007.38

Galliers, R. D., & Huang, J. C. (2012). The teaching of qualitative research methods in information systems: An explorative study utilizing learning theory. *European Journal of Information Systems*, *21*(2), 119–134. https://doi.org/10.1057/ejis.2011.44

Gardner, J. W. (2007). The Nature of Leadership. In *The Jossey-Bass Reader on Educational Leadership* (2nd ed.). Jossey-Bass.

Gardner, W. L., Lowe, K. B., Moss, T. W., Mahoney, K. T., & Cogliser, C. C. (2010). Scholarly leadership of the study of leadership: A review of The Leadership Quarterly’s second decade, 2000–2009. *Leadership Quarterly*, *21*(6), 922–958.

Ghawe, A., & Brohman, K. (2016). CIO Leadership Characteristics and Styles. *Americas Conference on Information Systems*, 1–10.

Ghobadi, S., & Mathiassen, L. (2016). Perceived barriers to effective knowledge sharing in agile software teams. *Information Systems Journal*, *26*(2), 95–125. https://doi.org/10.1111/isj.12053

Goldkuhl, G. (2012). Pragmatism vs interpretivism in qualitative information systems research. *European Journal of Information Systems*, *21*(2), 135–146. https://doi.org/10.1057/ejis.2011.54

Graham, C. M., Daniel, H., & Doore, B. (2015). Millennial Leadership: The oppositional relationship between leadership type and the quality of database system’s development in virtual environments. *International Journal of E-Collaboration*, *11*(3), 29–48. https://doi.org/10.4018/ijec.2015070103

Greenwood, R., & Lawrence, T. B. (2005). The Iron Cage in the Information Age: The Legacy and Relevance of Max Weber for Organization Studies. Editorial. *Organization Studies*, *26*(4), 493–499. https://doi.org/10.1177/0170840605051465

Guest, G., Bunce, A., & Johnson, L. (2006). How Many Interviews Are Enough ? An Experiment with Data Saturation and Variability. *Field Methods*, *18*(1), 59–82. https://doi.org/10.1177/1525822X05279903

Guhr, N., Lebek, B., & Breitner, M. H. (2019). The impact of leadership on employees’ intended information security behaviour : An examination of the full‐range leadership theory. *Information Systems Journal*, *29*, 340–362. https://doi.org/10.1111/isj.12202

Gumusluoglu, L., & Ilsev, A. (2009). Transformational leadership, creativity, and organizational innovation. *Journal of Business Research*, *62*(4), 461–473. https://doi.org/10.1016/j.jbusres.2007.07.032

Gupta, M., George, J. F., & Xia, W. (2019). Relationships between IT department culture and agile software development practices : An empirical investigation. *International Journal of Information Management*, *44*, 13–24. https://doi.org/10.1016/j.ijinfomgt.2018.09.006

Hickman, L., & Akdere, M. (2017). Development of Effective IT Leadership Behaviors : A Review. In *12th Midwest Association for Information Systems Conference*. Springfield.

Highsmith, J. (2001). History: The Agile Manifesto. Retrieved August 31, 2015, from http://www.agilemanifesto.org/history.html

Highsmith, J. (2002). *Agile Software Development Ecosystems*. Boston: Addison-Wesley.

Highsmith, J., & Cockburn, A. (2001). Agile Software Development: The Business of Innovation. *Computer*, *34*(9), 120–122.

Highsmith, J., Cockburn, A., Mellor, S. J., Jacobson, I., Henderson-Sellers, B., & Simons, M. (2002). The Great Methodologies Debate : Part 2. *Cutter IT Journal*, *15*(1), 1–40.

Highsmith, J., Sutherland, J., Glass, R. L., Moss, L. T., Kruchten, P., Wagner, L., & Russell, L. (2002). The Great Methodologies Debate: Part 1. *Journal of Information Technology Management*, *14*(12), 1–44.

Hill, N. S., Seo, M.-G., Kang, J. H., & Taylor, M. S. (2012). Building Employee Commitment to Change Across Organizational Levels : The Influence of Hierarchical Distance and Direct Managers ’ Transformational Leadership. *Organization Science*, *23*(3), 758–777.

Hines, P. A., Holweg, M., & Rich, N. L. (2004). Learning to Evolve: A Review of Contemporary Lean Thinking. *International Journal of Operations & Production Management*, *24*(10), 994–1011. https://doi.org/10.1108/01443570410558049

Hirschheim, R. A., Klein, H. K., & Lyytinen, K. (1995). *Information Systems Development and Data Modeling. Conceptual and Philosophical Foundations*. Cambridge: Cambridge University Press.

Hovorka, D. S., & Larsen, K. R. (2006). Enabling agile adoption practices through network organizations. *European Journal of Information Systems*, *15*, 159–168. https://doi.org/10.1057/palgrave.ejis.3000606

Hummel, M., Rosenkranz, C., & Holten, R. (2015). The role of social agile practices for direct and indirect communication in information systems development teams. *Communications of the Association for Information Systems*, *36*(15), 273–300. https://doi.org/10.17705/1CAIS.03615

Ichniowski, C., Shaw, K., & Crandall, R. W. (1995). Old Dogs and New Tricks: Determinants of the Adoption of Productivity-Enhancing Work Practices. *Brookings Papers on Economic Activity. Microeconomics*, *1995*, 1–65. https://doi.org/10.2307/2534771

Iivari, J., & Iivari, N. (2011). The relationship between organizational culture and the deployment of agile methods. *Information and Software Technology*, *53*(5), 509–520. https://doi.org/10.1016/j.infsof.2010.10.008

Iivari, J., & Maansaari, J. (1998). The usage of systems development methods: Are we stuck to old practices? *Information and Software Technology*, *40*(9), 501–510. https://doi.org/10.1016/S0950-5849(98)00077-9

Introna, L. D., & Whitley, E. A. (1997). Against Method-ism Exploring the limits of Method. *Information Technology & People*, *10*(1), 31–45.

Jiang, L., & Eberlein, A. (2009). An analysis of the history of classical software development and agile development. In *IEEE International Conference on Systems, Man and Cybernetics* (pp. 3733–3738). San Antonio, Texas. https://doi.org/10.1109/ICSMC.2009.5346888

Joshi, A., Lazarova, M. B., & Liao, H. (2009). Getting Everyone on Board: The Role of Inspirational Leadership in Geographically Dispersed Teams. *Organization Science*, *20*(1), 240–252. https://doi.org/10.1287/Orsc.1080.0383

Kaplan, B., & Duchon, D. (1988). Combining Qualitative and Quantitative Information Systems: A case study. *MIS Quarterly*, *12*(4), 571–586.

Kaplan, B., & Maxwell, J. A. (1994). Qualitative Research Methods for Evaluating Computer Information Systems. *Evaluation Health Care Information Systems: Methods and Application*.

Karahanna, E., & Watson, R. T. (2006). Information Systems Leadership. *IEEE Transactions on Engineering Management*, *53*(2), 171–176. https://doi.org/10.1109/TEM.2006.872247

Kerr, J. (2013). *Legacy: 15 Lessons in Leadership*. London: Constable & Robinson.

Kezar, A. (2004). Philosophy, Leadership, and Scholarship: Confucian Contributions to a Leadership Debate. *Leadership Review*, *4*(1), 110–131.

Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, *23*(1), 67–94.

Korhonen, K. (2013). Evaluating the impact of an agile transformation: A longitudinal case study in a distributed context. *Software Quality Journal*, *21*(4), 599–624. https://doi.org/10.1007/s11219-012-9189-4

Kose, M. A., & Ozturk, E. O. (2014). A World of Change: Taking stock of the past half century. *Finance & Development*, *51*(3), 6–11.

Kostova, T., & Roth, K. (2002). Adoption of an Organizational Practice By Subsidiaries of Multinational Corporations: Institutional and Relational Effects. *Academy of Management Journal*, *45*(1), 215–233. https://doi.org/10.2307/3069293

Kurapati, N., Manyam, V. S. C., & Petersen, K. (2012). Agile software development practice adoption survey. In *International Conference on Agile Software Development* (pp. 16–30). Berlin, Heidelberg: Springer.

Kurtzman, J. (2010). *Common Purpose: How Great Leaders Get Organizations to Achieve the Extraordinary* (1st ed.). San Francisco: Jossey-Bass.

Lang, M., Keaveney, S., & Conboy, K. (2013). Cost Estimation in Agile Software Development Projects. In R. Pooley, J. Coady, C. Schneider, H. Linger, C. Barry, & M. Lang (Eds.), *Information Systems Development*. New York, NY: Springer.

Layman, L., Williams, L., & Cunningham, L. (2004). Motivations and measurements in an agile case study. *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, *05*-*Novembe*, 14–24. https://doi.org/10.1145/1151433.1151436

LeCompte, M., & Goetz, J. P. (1982). Problems of Reliability and Validity in Ethnographic Research. *Review of Educational Research*, *52*(1), 31. https://doi.org/10.2307/1170272

Lee, D., Sambamurthy, V., Lim, K. H., & Wei, K. K. (2015). How Does IT Ambidexterity Impact Organizational Agility? *Information Systems Research*, *26*(2), 398–417. https://doi.org/10.1287/isre.2015.0577

Lee, G., & Xia, W. (2010). Toward agile: An integrated analysis of quantitative and qualitative field data on software development agility. *MIS Quarterly*, *34*(1), 87–114. https://doi.org/Article

Lee, S., & Yong, H.-S. (2013). Agile Software Development Framework in a Small Project Environment. *Journal of Information Processing Systems*, *9*(1), 69–88. https://doi.org/http://dx.doi.org/10.1108/17506200710779521

Leroy, F., & Ramanantsoa, B. (1997). The cognitive and behavioural dimensions of organizational learning in a merger: An empirical study. *Journal of Management Studies*, *34*(6), 871–894. https://doi.org/10.1111/1467-6486.00076

Li, C., Zhao, H., & Begley, T. M. (2015). Transformational leadership dimensions and employee creativity in China: A cross-level analysis. *Journal of Business Research*, *68*, 1149–1156. https://doi.org/10.1016/j.jbusres.2014.11.009

Li, Y., Tan, C. H., & Teo, H. H. (2012). Leadership characteristics and developers’ motivation in open source software development. *Information and Management*, *49*, 257–267. https://doi.org/10.1016/j.im.2012.05.005

Lohan, G., Conboy, K., & Lang, M. (2010). Beyond budgeting and agile software development: A conceptual framework for the performance management of agile software development teams. *ICIS 2010 Proceedings - Thirty First International Conference on Information Systems*.

Lowe, K. B., Kroeck, K. G., & Sivasubramaniam, N. (1996). Effectiveness correlates of transformational and transactional leadership : A meta-analytic review of the MLQ literature. *Leadership Quarterly*, *7*(3), 385–425.

Lowry, P. B., & Wilson, D. (2016). Creating Agile Organizations through IT: The Influence of Internal IT Service Perceptions on IT Service Quality and IT Agility. *Journal of Strategic Information Systems*, *25*, 211–226. https://doi.org/10.1016/j.jsis.2016.05.002

Lu, Y., & Ramamurthy, K. R. (2011). Understanding the link between Information Technology Capability and Organizational Agility: An Empirical Examination. *MIS Quarterly*, *35*(4), 931–954.

Lycett, M., Macredie, R. D., Patel, C., & Paul, R. J. (2003). Migrating Agile Methods to Standardized Development Practice. *Computer*, (June), 79–85.

Markus, M. L. (1989). Case Selection in a Disconfirmatory Case Study. In *The Information Systems Research Challenge, Harvard Business School Research Colloquium* (pp. 20–26). Boston: Harvard Business School.

Maskell, B. (2001). Insight from industry: The age of agile manufacturing. *Supply Chain Management: An International Journal*, *6*(1), 5–11.

Mason, M. (2010). Sample Size and Saturation in PhD Studies Using Qualitative Interviews. *Qualitative Social Research*, *11*(3:8).

McHugh, O., Conboy, K., & Lang, M. (2011). Using agile practices to influence motivation within IT project teams. *Scandinavian Journal of Information Systems*, *23*(2), 59–84.

McHugh, O., Conboy, K., & Lang, M. (2012). Agile practices: The impact on trust in software project teams. *IEEE Software*, *29*(3), 71–76. https://doi.org/10.1109/MS.2011.118

Mencke, R. (2008). A product manager’s guide to surviving the big bang approach to agile transitions. In *Agile 2008* (pp. 407–412). https://doi.org/10.1109/Agile.2008.65

Meso, P., & Jain, R. (2006). Agile software development: Adaptive systems principles and best practices. *Information Systems Management*, *23*(3), 19–30. https://doi.org/10.1201/1078.10580530/46108.23.3.20060601/93704.3

Miles, M. B., Huberman, A. M., & Saldaña, J. (2014). *Qualitative Data Analysis* (3rd ed.). London: Sage.

Mishra, A., Garbajosa, J., Wang, X., Bosch, J., & Abrahamsson, P. (2017). Future Directions in Agile Research: Alignments and Divergence between Research and Practice. *Journal of Software: Evolution and Process*, *29*(e1884). https://doi.org/10.1002/smr.1884

Misra, S., Kumar, V., Kumar, U., Fantazy, K., & Akhter, M. (2012). Agile software development practices: Evolution, principles, and criticisms. *International Journal of Quality & Reliability Management*, *29*(9), 972–980. https://doi.org/10.1108/02656711211272863

Moe, N. B., Dingsøyr, T., & Dybå, T. (2008). Understanding self-organizing teams in agile software development. In *19th Australian Conference on Software Engineering* (pp. 76–85). https://doi.org/10.1109/ASWEC.2008.4483195

Myers, M. D. (1997). Qualitative research in information systems. *MIS Quarterly*, (June).

Myers, M. D. (1999). Investigating Information Systems with Ethnographic Research. *Communications of the Association for Information Systems*, *2*(December). https://doi.org/10.17705/1cais.00223

Myers, M. D. (2009). *Qualitative Research in Business and Management* (1st ed.). London, England: Sage Publications.

Myers, M. D. (2013). *Qualitative Methods in Business and Management* (2nd ed.). Thousand Oaks, CA: Sage Publications.

Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. *Information and Organization*, *17*(1), 2–26. https://doi.org/10.1016/j.infoandorg.2006.11.001

Nandhakumar, J., & Avison, D. (1999). The fiction of methodological development: a field study of information systems development. *Information Technology & People*, *12*(2), 176–191. https://doi.org/10.1108/09593849910267224

Nazir, S., & Pinsonneault, A. (2012). IT and Firm Agility : An Electronic Integration Perspective. *Journal of the Association for Information Systems*, *13*(3), 150–171.

Ola, B. (2017). A Critique of Conceptual Leadership Styles’. In P. Godbole, D. Burke, & J. Aylotte (Eds.), *Why Hospitals Fail* (pp. 57–68). Springer.

Pauleen, D. J. (2003). An Inductively Derived Model of Leader-Initiated Relationship Building with Virtual Team Members. *Journal of Management Information Systems*, *20*(3), 227–256. https://doi.org/10.1080/07421222.2003.11045771

Pfeffer, J. (1997). *New directions for Organization Theory - Problems and Prospects*. New York; Oxford: Oxford Universiry Press. https://doi.org/10.1017/CBO9781107415324.004

Pil, F. K., & MacDuffie, J. P. (1996). The Adoption of High-Involvement Work Practices. *Industrial Relations: A Journal of Economy and Society*, *35*(3), 423–455. https://doi.org/10.1111/j.1468-232X.1996.tb00414.x

Podsakoff, P. M., Mackenzie, S. B., Moorman, R. H., & Fetter, R. (1990). Transformational Leader Behaviors and Their Effects on Followers’ Trust in Leader, Satisfaction, and Organizational Citizenship Behaviors. *Leadership Quarterly*, *1*(2), 107–142.

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development*. Addison-Wesley.

Qu, R., Janssen, O., & Shi, K. (2015). Transformational leadership and follower creativity: The mediating role of follower relational identification and the moderating role of leader creativity expectations. *Leadership Quarterly*, *26*, 286–299. https://doi.org/10.1016/j.leaqua.2014.12.004

Ramesh, B., Mohan, K., & Cao, L. (2012). Ambidexterity in Agile Distributed Development: An Empirical Investigation. *Information Systems Research*, *23*(2), 323–339.

Ramirez, M., Guy, F., & Beale, D. (2007). Contested resources: unions, employers, and the adoption of new work practices in US and UK telecommunications. *British Journal of Industrial Relations*, *45*(3), 495–517.

Riemenschneider, C. K., Hardgrave, B. C., & Davis, F. D. (2002). Explaining software developer acceptance of methodologies: A comparison of five theoretical models. *IEEE Transactions on Software Engineering*, *28*(12), 1135–1145.

Rost, J. C. (1991). *Leadership for the twenty-first century*. New York: Praeger.

Rubin, R. S., Munz, D. C., & Bommer, W. H. (2005). Leading from within: The Effects of Emotion Recognition and Personality on Transformational Leadership Behavior. *The Academy of Management Journal*, *48*(5), 845–858. https://doi.org/10.5465/AMJ.2005.18803926

Russo, N. L., & Stolterman, E. (2000). Exploring the assumptions underlying information systems methodologies. *Information Technology & People*, *13*(4), 313–327. https://doi.org/10.1108/09593840010359509

Saldaña, J. (1987). An Introduction to Codes and Coding. In *The Coding Manual for Qualitative Researchers* (pp. 1–31). https://doi.org/10.1519/JSC.0b013e3181ddfd0a

Sambamurthy, V., Bharadwaj, A., & Grover, V. (2003). Shaping Agility through Digital Options: Reconceptualizing the Role of Information Technology in Contemporary Firms. *MIS Quarterly*, *27*(2), 237–236. https://doi.org/10.2307/30036530

Sandberg, J. (2005). How do we justify knowledge produced within interpretive approaches? *Organizational Research Methods*, *8*(1), 41–68. https://doi.org/10.1177/1094428104272000

Sawyer, S. (2004). Software Development Teams. *Communications of the ACM*, *47*(12), 95–99.

Schwaber, K., & Sutherland, J. (2017). The Scrum Guide. *Scrum.Org*.

Shao, Z., Feng, Y., & Hu, Q. (2016). Effectiveness of top management support in enterprise systems success: a contingency perspective of fit between leadership style and system life-cycle. *European Journal of Information Systems*, *25*(2), 131–153. https://doi.org/10.1057/ejis.2015.6

Shao, Z., Feng, Y., & Liu, L. (2012). The mediating effect of organizational culture and knowledge sharing on transformational leadership and Enterprise Resource Planning systems success: An empirical study in China. *Computers in Human Behavior*, *28*(6), 2400–2413. https://doi.org/10.1016/j.chb.2012.07.011

Sharma, S., & Rai, A. (2015). Adopting IS process innovations in organizations: the role of IS leaders’ individual factors and technology perceptions in decision making. *European Journal of Information Systems*, *24*, 23–37. https://doi.org/10.1057/ejis.2013.24

Sidky, A., Arthur, J., & Bohner, S. (2007). A disciplined approach to adopting agile practices: The agile adoption framework. *Innovations in Systems and Software Engineering*, *3*(3), 203–216. https://doi.org/10.1007/s11334-007-0026-z

Solinski, A., & Petersen, K. (2016). Prioritizing agile benefits and limitations in relation to practice usage. *Software Quality Journal*, *24*(2), 447–482. https://doi.org/10.1007/s11219-014-9253-3

Soundararajan, S., Arthur, J., & Balci, O. (2012). A Methodology for Assessing Agile Software Development Approaches. In *2012 Agile Conference, Agile 2012* (pp. 51–54). https://doi.org/10.1109/Agile.2012.24

Starks, H., & Trinidad, S. B. (2007). Choose your method: a comparison of phenomenology, discourse analysis, and grounded theory. *Qualitative Health Research*, *17*(10), 1372–1380. https://doi.org/10.1177/1049732307307031

Staron, M. (2006). Adopting Model Driven Software Development in Industry – A Case Study at Two Companies. In O. Nierstrasz, J. Whittle, D. Harel, & G. Reggio (Eds.), *International Conference on Model Driven Engineering Languages and Systems* (pp. 57–72). Berlin, Heidelberg: Springer. https://doi.org/10.1007/11880240\_5

Strode, D. E. (2016). A dependency taxonomy for agile software development projects. *Information Systems Frontiers*, *18*(1), 23–46. https://doi.org/10.1007/s10796-015-9574-1

Sutherland, J. (2014). *Scrum: The Art of Doing Twice as Much in Half the Time*. New York: Crown Business.

Tallon, P. P., & Pinsonneault, A. (2011). Competing Perspectives on the Link Between Strategic Information Technology Alignment and Organizational Agility: Insights from a Mediation Model. *MIS Quarterly*, *35*(2), 463–486.

Tallon, P. P., Queiroz, M., Coltman, T., & Sharma, R. (2019). Information technology and the search for organizational agility: A systematic review with future research possibilities. *Journal of Strategic Information Systems*, *28*, 218–237. https://doi.org/10.1016/j.jsis.2018.12.002

Teddlie, C., & Tashakkori, A. (2012). Common “Core” Characteristics of Mixed Methods Research: A Review of Critical Issues and Call for Greater Convergence. *American Behavioral Scientist*, *56*(6), 774–788. https://doi.org/10.1177/0002764211433795

Tolfo, C., & Wazlawick, R. S. (2008). The influence of organizational culture on the adoption of extreme programming. *Journal of Systems and Software*, *81*, 1955–1967. https://doi.org/10.1016/j.jss.2008.01.014

Tripp, J. F., & Armstrong, D. J. (2018). Agile Methodologies: Organizational Adoption Motives, Tailoring, and Performance. *Journal of Computer Information Systems*, *58*(2), 170–179. https://doi.org/10.1080/08874417.2016.1220240

Tripp, J. F., Riemenschneider, C. K., & Thatcher, J. B. (2016). Job Satisfaction in Agile Development Teams: Agile Development as Work Redesign. *Journal of the Association for Information Systems*, *17*(4), 267–307.

Truex, D., Baskerville, R., & Travis, J. (2000). Amethodical systems development: The deferred meaning of systems development methods. *Accounting, Management and Information Technologies*, *10*(1), 53–79. https://doi.org/10.1016/S0959-8022(99)00009-0

Tsang, E. W. K. (1997). Organizational Learning and the Learning Organization: A Dichotomy Between Descriptive and Prescriptive Research. *Human Relations*, *50*(1), 73–89.

Turan, S., & Sny, C. L. (1996). An exploration of transformational leadership and its role in strategic planning: A conceptual framework. In *Annual Meeting of the International Society for Educational Planning*. New Orleans.

Van Knippenberg, D., & Sitkin, S. B. (2013). A Critical Assessment of Charismatic—Transformational Leadership Research: Back to the Drawing Board? *The Academy of Management Annals*, *7*(1), 1–60. https://doi.org/10.1080/19416520.2013.759433

Version One. (2017). *10th Annual State of Agile Survey 2017*.

Version One. (2018). *12th Annual State of Agile Report*.

Vidgen, R., & Wang, X. (2009). Coevolving Systems and the Organization of Agile Software Development. *Information Systems Research*, *20*(3), 355–376. https://doi.org/10.1287/isre.1090.0237

Walsham, G. (2006). Doing interpretive research. *European Journal of Information Systems*, *15*(3), 320–330. https://doi.org/10.1057/palgrave.ejis.3000589

Wang, X., Conboy, K., & Cawley, O. (2012). “Leagile” software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of Systems and Software*, *85*(6), 1287–1299. https://doi.org/10.1016/j.jss.2012.01.061

Wastell, D. G. (1996). The Fetish of Technique: Methodology as a Social Defence. *Information Systems Journal*, *6*(1), 25–40.

Williams, L. (2012). What agile teams think of agile principles. *Communications of the ACM*, *55*(4), 71. https://doi.org/10.1145/2133806.2133823

Williams, L., Rubin, K., & Cohn, M. (2010). Driving process improvement via Comparative Agility assessment. *Proceedings - 2010 Agile Conference, AGILE 2010*, 3–10. https://doi.org/10.1109/AGILE.2010.12

Wolcott, H. F. (1994). *Transforming Qualitative Data: Description, Analysis, and Interpretation*. Thousand Oaks, CA: Sage Publications.

Xu, P., & Shen, Y. (2015). Leading Agile Teams: An Exploratory Study of Leadership Styles in Agile Software Development. In *21st Americas Conference on Information Systems* (pp. 1–7).

Xu, P., & Shen, Y. (2016). Empowering Leadership, Transactive Memory Systems and Agility in Software Development Teams: A Theoretical Framework. In *22nd Americas Conference on Information Systems 2016* (pp. 1–10). San Diego: AMCIS.

Xu, P., & Shen, Y. (2018). The Role of Leadership in Agile Software Development. In *International Research Workshop on IT Project Management 2018*.

Yin, R. K. (2013). Validity and generalization in future case study evaluations. *Evaluation*, *19*(3), 321–332. https://doi.org/10.1177/1356389013497081

Yin, R. K. (2014). *Case Study Research: Design and Methods* (5th ed.). London: Sage.

Yukl, G. (1989). *Leadership in Organizations* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.

Yukl, G. (2012). Effective Leadership Behavior: What We Know and What Questions Need More Attention. *Academy of Management Perspectives*, *26*(4), 66–85. https://doi.org/10.5465/amp.2012.0088

Zhang, X., & Bartol, K. M. (2010). Linking empowering leadership and employee creativity: the influence of psychological empowerment, intrinsic motivation, and creative process engagement. *Academy of Management Journal*, *53*(1), 107–128. https://doi.org/10.5465/AMJ.2010.48037118

Zimbardo, P. G., Ebbesen, E. B., & Maslach, C. (1977). *Inuencing Attitudes and Changing Behavior*. Reading: Addison-Wesley.

# Appendix

This appendix includes copies of interview protocols, consent forms and other reference materials which demonstrate the procedures and process of this study in greater detail.

## ../../../Downloads/Untitled_message/Interview_ParticipationEmail.pngEmail inviting participants to interview

## Informed Consent and Demographics Form

**Interview Consent**

This interview is conducted in the context of a PhD with Loughborough University on the subject of Organisational Agility, as such any information you provide will be treated as confidential and will only be used in relation to this study.

*I have read and understood this statement (please tick)*

Your participation in this interview is voluntary and you can choose to stop at any time. In advance of the interview please fill out as much of the background information on this page as possible. Any data used for publication from will be anonymised.

*I have read and understood this statement (please tick)*

If at any time during the interview any questions are unclear, please do not hesitate to ask for clarification. There are no right or wrong answers to the questions posed, the aim is to gather data to better understand how members of an agile software development team experience agile principles and practices within their teams and the wider organisation. The focus is on your personal experience of agile principles and practices in your current environment.

If you agree to the interview being recorded you can, at any time, ask to pause or cease recording. The purpose of recording the interview is to produce a transcript which will be made available to you, all recordings and transcripts will be considered confidential. Any data used for publication will be anonymised.

*I have read and understood this statement (please tick)*

*I am happy for the interview to be recorded (please tick)*

**Background Information**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Where multiple options are given, please circle the most appropriate answer | | | | | | |
| **Name** |  | | | | | |
| **Job Title** |  | | | | | |
| **Age** | Under 25 | 25-34 | 35-44 | 45-54 | 55-64 | 65 or over |
| **What is your highest level of education** | GCSE  (or equiv.) | A Level  (or equiv.) | Degree | Masters | PhD | Other |
| **Number of years employed at current organisation** | Less than 1 Year | 1-3 Years | 3-5 Years | 5-7 Years | 7-9 Years | More than 9 Years |
| **Number of years *using* agile methods** | Less than 1 Year | 1-3 Years | 3-5 Years | 5-7 Years | 7-9 Years | More than 9 Years |
| **Number of years *knowledge* of agile methods** | Less than 1 Year | 1-3 Years | 3-5 Years | 5-7 Years | 7-9 Years | More than 9 Years |

## Interview Protocol

Interview Guide (Tech)

1. How would you describe your role at SoftwareCorps?

* What are your main responsibilities?
* What skills do you need to carry out your role?

1. Tell me a bit about the team you are a member of within your department.

* What products do you work on?
* What different roles and skills make up the team?

1. There have been a number of changes in the way Technology work to deliver software over the past few years.

* Knowing how you work now, can you describe the old way of working?
* What are the biggest differences between the “old” way and how you work now? Can you give examples?

1. How would you define agility in terms of software development, in your own words?

* What makes software development “agile”?
* Based on your own definition of Agile, can you give examples of where and how Technology has achieved this so far?
* Are there any principles or practices you think Technology have not established yet?
  + Why have these not been established?
  + What were the barriers?
  + How could these barriers be overcome?

1. Could you summarise the agile practices you carry out as a member of an agile software development team, and give some examples?

* How do each of these practices help the team to be agile?
* How are these practices different to how you worked before?
* Can you give examples?

1. In your opinion, are agile principles and practices embedded in the team?

* How do you know?
* How do you determine, as a team, which agile practices to follow?
* To what extent would you say that you are typically free to organize your work and the practices you follow?
* What is stopping your team going back to the old way of working?
* Do you think things could revert to the old way?
* Would you want to go back to the old way of working? What makes you say that?

1. Thinking of departments outside the Technology team, how agile do you think the organisation is?

* Why do you say that?
* Do you think other parts of the organisation would like to adopt agile practices?
* Do you think they could?
* Do you think there are any barriers or facilitators to other parts of the organisation adopting agile practices?
* Do you think that a lack of agile practices in the rest of the organisation would impact on your teams ability to use and adoption of agile practices? If so, why?
* How would you define Agility in terms of the organisation?
* Are there any key agile practices or principles that you think SoftwareCorps is yet to establish?
  + Are there any particular areas you would focus on addressing first?

Interview Guide (CTO)

1. Can you tell me a bit about your background before SoftwareCorps?
   * Where have you worked?
   * What are your expertise?
2. How did you come to work at SoftwareCorps?
   * In what capacity did you join SoftwareCorps?
   * What did you do as part of this role?
3. Can you describe the way the Technology function worked before you became CTO?
   * What worked well?
   * What didn’t work well?
4. How would you define agility in terms of software development?
   * What principles and practices make SD agile?
   * Would you describe the Technology function as agile?
     1. Can you give examples?
   * What agile principles and practices are followed?
     1. Can you give examples?
   * Are there any principles or practices you think Technology have not established yet?
     1. Why have these not been established?
     2. What were the barriers?
     3. How could these barriers be overcome?
5. How would you define agility in terms of the organisation?
   * Do you consider SoftwareCorps and agile organisation?
   * Are there any barriers to becoming more agile?
   * Can you give examples?
   * How could these be overcome?
6. How did you come to be CTO?
7. What were you asked to do in your role as CTO?
   * What does “improve productivity” mean?
   * What, if any, changes have you introduces so far and what is the aim?
   * What, if any, changes are you planning to introduce and what is the aim?

Interview Guide (PO)

1. How would you describe your role at SoftwareCorps?

* What are your main responsibilities?
* What skills do you need to carry out your role?

1. Tell me a bit about the team you are a member of

* What different roles and skills make up the team?

1. There have been a number of changes in the way Technology and Product work together to deliver software over the past few years.

* Knowing how you work now, can you describe the old way of working?
* What are the biggest differences between the “old” way and how you work now? Can you give examples?

1. How would you define agility in terms of software development, in your own words?

* What makes software development “agile”?
* Based on your own definition of Agile, can you give examples of where and how Product and Technology have achieved this so far?
* Are there any principles or practices you think have not been established yet?
  + Why have these not been established?
  + What were the barriers?
  + How could these barriers be overcome?

1. Could you summarise the agile practices you carry out as a member of an agile software development team, and give some examples of how these were used?

* How do each of these practices help the team to be agile?
* How are these practices different to how you worked before?
* Can you give examples?

1. In your opinion, are agile principles and practices embedded in the team?

* How do you know?
* How do you determine, as a team, which agile practices to follow?
* To what extent would you say that you are typically free to organize your work and the practices you follow?
* What is stopping your team going back to the old way of working?
* Do you think things could revert to the old way?
* Would you want to go back to the old way of working? What makes you say that?

1. Thinking of departments outside of Technology, how agile do you think the organisation is?

* Why do you say that?
* Do you think other parts of the organisation would like to adopt agile practices?
* Do you think they could?
* Do you think there are any barriers or facilitators to other parts of the organisation adopting agile practices?
* Do you think that a lack of agile practices in the rest of the organisation would impact on your teams ability to use and adoption of agile practices? If so, why?
* Are there any barriers present in the organisation which impact on your ability to deliver software successfully?
* How would you define Agility in terms of the organisation?
* Are there any key agile practices or principles that you think SoftwareCorps is yet to establish?
  + Are there any particular areas you would focus on addressing first?

Interview Guide (COO)

1. Can you tell me a bit about your role at SoftwareCorps?
2. Before the creation of the CTO role, how would you describe the development of software at SoftwareCorps?
   * E.g. frequency of delivery of software? Visibility of progress and priorities?
3. What were the motivations for creating a CTO role?
4. What were you looking for in a CTO initially?
5. As the first CTO, how would you describe Jamie’s approach?
   * What were your expectations of him in this new role?
   * What was he asked to do by the business?
   * Were there any barriers to the changes he introduced?
   * Were there any facilitators to the changes he introduced?
     1. E.g. Embedding new habits? Learning? Leadership?
6. When Jamie left Nelson stepped in, temporarily at first, before taking on the role officially.
   * What were the motivations behind giving Nelson that role?
   * How would you describe his approach as CTO?
   * How did it differ from that of Jamie?
   * What were your expectations of him in this role?
   * What was he asked to do by the business?
   * Were there any barriers to the changes he introduced?
   * Were there any facilitators to the changes he introduced?
7. Considering how you described SoftwareCorps’ development approach before the creation of the CTO role, how would you describe our development approach now?
8. From your perspective as COO, what does it mean to be Agile?
   * What were the motivations for seeking a more Agile development method?
   * Were there any wider business pressures or strategic aims that encouraged the need for Agility in the business?
   * Have business pressures or strategy changed since we began working on becoming more Agile? What are they now?
9. When Nelson left, what were you looking for in his replacement?
10. How would you describe Julius’ approach as CTO?
    * How does it differ from that of Nelson?
    * What are your expectations of him in this role?
    * What has he been asked to do by the business?

Interview Guide (Leadership)

1. Can you start by telling me how you would describe LEADERS approach as a leader?

* What was/is he like to work with?

1. **(II)** How would you describe LEADERS approach to teamwork?

* *e.g. Does/did he discourage teamwork? foster collaboration, trust and cooperation, encourage you to be a team player, develop team spirit among employees?*
* *e.g. Does/did he emphasise the importance of going beyond self-interests for the good of the group?*
* Does LEADERS approach to teamwork affect how you approach ASD? If so, how?

1. **(II)** In your experience, does LEADER have any values or beliefs that are important to him?

* If so, what were/are these? (e.g. integrity? trustworthy?, honest and open? moral? ethical?)
* Does he practice what he preaches? If so, can you give any examples? (makes decisions based on values and beliefs?)
* Do LEADERS values affect how you approach ASD? If so, how?

1. **(IM)** Does/did LEADER communicate the goals for the future? If so,

* How is that communicated? (clear? positive? Enthusiastic? Interesting? Inspiring? Compelling? Face to face?)
* *E.g. Does/did he specify the importance of having a strong sense of purpose?*
* *E.g. Does/did he get the group to work together for the same goal?*
* Can you give examples?
* How does/did LEADER get buy in for their goals?
* *E.g. express confidence in your ability to achieve them?*
* Do LEADERS goals for the future affect how you approach ASD? If so, how?

1. **(CR)** How did LEADER approach setting performance targets and expectations?

* *E.g. discuss in specific terms who is responsible for achieving targets?*
* *E.g. expects a lot from you? Insists on the best? Won’t settle for second best?*
* (**CR**) How does/did LEADER act when you performed well/ your work was very good?
* *e.g. personally compliments you? Gives positive feedback to you? Gives you special recognition? Commends you? Express satisfaction? Does not acknowledge?*
* **(ME)** How does/did LEADER deal with mistakes/deviations from standards/complaints in subordinates?
* E.g. focus attention on them? Wait until become serious? Wait for things to go wrong? Keeps track of all mistakes? Focus full attention on them? Avoids getting involved? Avoids making decisions?
* **(IM)** Would you say you were given recognition/encouragement by LEADER? If so, how (examples)?
* **(CR)** How does/did LEADER approach reward and remuneration?
* *e.g. provide assistance in return for effort? Make it clear what you can expect if goals achieved?*
* Does LEADERS approach to your performance impact how you approach ASD? If so, how?

1. **(II)** Does/did LEADERS approach impact how you feel?

* If so, how? E.g. does he instil you with pride? Feel respected? Inspire you to be highly competent?
* **(IC)** Does/did LEADER respect your personal feelings?
* If so, how? Examples?
* **(IC)** Does/did LEADER treat people as individuals as opposed to just a member of a group?
* E.g. consider you as having different needs, abilities and aspirations
* If so, how? Examples?
* Does LEADERS approach to you individually impact how you approach ASD? If so, how?

1. **(II)** Would you say LEADER leads by example?

* E*.g. leads by doing rather than simply telling?*
* *E.g. provides a good model to follow?*
* *E.g. acts in a way to build respect?*
* **(IC)** Does/did LEADER spend time teaching and coaching you individually?
* E.g. approachable, help develop your strengths
* If so, how? Examples?
* Does the example set by LEADER impact how you approach ASD? If so, how?

1. **(IS)** How does/did LEADER get you to approach problem-solving?

* E.g. challenge assumptions? Think of old problems in new ways? Encourage strategic vs short term thinking? Provide novel solutions to difficult problems?
* E.g. encouraged to participate in critical thinking? Seeks your input in solution building?
* Does LEADERS approach to problem-solving impact how you approach ASD? If so, how?

1. **(IC)** Does/did LEADER involve others when making decisions?

* If so, how? Examples?
* Does LEADERS approach to decision making impact how you approach ASD? If so, how?

## Overview of Findings in Leadership Literature

| **Theme** | **Papers** | **Level of Leader** | **Findings** |
| --- | --- | --- | --- |
| Rollout and adoption of information systems | Cho et al. 2011 | Various levels | Tf leadership enhances IS usage |
| *Shao et al. 2012* | *Senior Leadership* | *Diff. leader styles (tf/tx) suit diff. phases of ES adoption lifecycle.* |
| Sharma and Rai 2015 | Executives | Individual factors (e.g. perception of tech) differentiate adopter/non- adopter |
| *Shao et al. 2016* | *Top Management* | Transformational leaders promote desired org. culture and knowledge sharing intention which impact ERP success |
| Team Behaviours | Faraj and Sambamurthy 2006 | Project Leader | Empowering leadership suits experienced teams with high task uncertainty |
| *Gumusluoglu and Ilsev 2009* | *Supervisors* | *Tf style positively influences creativity* |
| Zhang and Bartol 2010 | Middle Managers | Tf style positively influences creativity. |
| *Li et al. 2012* | *Project Leader* | *Intrinsic/extrinsic motivation of devs indicates whether tf/tx style more appropriate*. |
| *Guhr et al. 2019* | Various | Transformational leaders capable of influencing extra‐role and in‐role behaviour levels |
| Dispersed Teams | Joshi et al. 2009 | Team Leader | Inspirational leaders more important in dispersed teams |
| *Graham et al. 2015* | *Team Leader* | *Tf leadership ineffective in dispersed teams* |
| Charlier et al. 2016 | Emergent within team | The co-location/dispersion of team members has a significant effect on leadership perceptions |
| *Eseryel and Eseryel 2013* | *Emergent within team* | *Practitioners need to be actively engaged in teams’ work in order to emerge as transformational leaders.* |
| Pauleen 2003 | Team Leader | Virtual leaders build relationships by: assessing conditions, targeting level of relationship, and creating strategies |
| Agile software development | Dubinsky and Hazzan 2010 | Change Leader | Ad-hoc leadership style most suited to agile transition. |
| *Xu and Shen 2015* | *Project Leader* | *Research framework to assess effect of leader style on project performance* |
| Xu and Shen 2016 | Team Leader | Empowering leadership enables team agility |
| *Xu and Shen 2018* | *Within team* | *Identify five leader roles and activities which affect learning* |
| Literature Review | Karahanna and Watson 2006 | CIO/TMT | Identified dominant themes in IS leadership lit. |
| *Ghawe and Brohman 2016* | *CIO* | *Four categories of CIO characteristics: fundamental, expertise, relationship, and sensing* |
| Hickman and Akdere 2017 | Various | Leader development req. formal mentoring, robust feedback, and should be a core process |

## Summary of definitions of the 4 I's of Transformational Leadership

| **Dimension>>** | **Individualised Consideration** | **Intellectual Stimulation** | **Inspirational Motivation** | **Idealised influence (Charisma)** |
| --- | --- | --- | --- | --- |
| **Reference** |
| **Transformational leadership dimensions and employee creativity in China: A cross-level analysis**  (C. Li et al., 2015) | The degree to which a leader acts as a mentor or coach and attends to each follower's needs and concerns.  Showing concern for individual employees and their families.  Chinese leaders attend not only to employee work-related needs and career development but also their non-work and family member needs. | The degree to which a leader stimulates activity through job autonomy, empowerment, and encouragement. | The degree to which a leader articulates an appealing vision to followers, communicates optimism about future goals, and provides meaning for the task at hand.  Vision articulation, based on providing employees with clear vision and direction | The degree to which a leader shows charisma and emphasizes the importance of purpose, commitment, and the ethical consequences of decisions.  Moral modelling, based on altruism, freedom from corruption, fairness, and hard work  Charisma, based on passion and courage |
| **Transformational leadership and follower creativity: The mediating role of follower relational identification and the moderating role of leader creativity expectations**  (Qu et al., 2015) | Model empathy and support for individual concerns and openness to new suggestions and approaches.  Followers may feel free to think in new ways, go beyond standard practices, and proceed with creativity without fear of penalties.  Treats followers individually and differently on the basis of their talents and knowledge. Pays attention to their individual concerns and needs, and acts as a mentor to encourage their personal growth and development…thereby engendering relational identification with the leader. | Promotes critical thinking and problem-solving in the follower.  Promote creativity by encouraging followers to question critical assumptions underlying the established framework of thoughts and routines and to look at old problems and situations in new ways. | None given | None given |
| **What makes leadership, leadership? Using self-expansion theory to integrate traditional and contemporary approaches**  (Van Knippenberg & Sitkin, 2013) | A recognition of the differing needs of followers, individualized attention, and coaching. | Stimulate problem-solving and careful and creative consideration of the issues at hand. | The communication of an inspiring vision and high-performance expectations.  Vision, mission, and collective sense of purpose, idealized influence and inspirational motivation are typically…combined into one “charisma” factor. | Known both as idealized influence and as charisma.  Instilling pride, respect, and trust, as well as the perception that the leader conveys a strong sense of purpose and collective mission. |
| **A Critical Assessment of Charismatic—Transformational Leadership Research: Back to the Drawing Board?**  (Dansereau et al., 2013) | Takes into account the individual needs of the follower in achieving these shared goals.  Leader shows concern and respect for followers, looks out for their welfare, and expresses appreciation and support.  Leader builds relationships with group members and encourages interpersonal relationships to achieve group goals. | Provides followers with the path to solve problems in new ways. | Persuading followers to adopt shared goals. | None given. |
| **Further specification of the leader political skill-leadership effectiveness relationships: Transformational and transactional leader behavior as mediators**  (Ewen et al., 2013) | Attending to the individual  differences of followers, by listening effectively, and by establishing a supportive climate.  Considering the individual needs of followers. | Leaders help make their followers more creative and innovative.  Persuade followers to think through situations prior to action and to think beyond the current and common conceptualizations.  Encourage followers to think beyond the initial outcome, and to plan for unexpected consequences of success  Generates an environment welcoming new ideas and different opinions. | Engage followers by crafting and communicating a compelling vision. | Emphasizing the greater good of and for the group. |
| **How does leadership affect information systems success? the role of transformational leadership**  (Cho et al., 2011) | The degree to which a leader provides support and encouragement to followers, coaches and mentors them.  Coach or mentor followers and provide individualized support while listening to the concerns and needs of IS users. | The degree to which a leader increases awareness and helps followers challenge assumptions and take risks.  Stimulate system users’ creative problem-solving skills by challenging them to address old problems using new perspectives, making them take risks, and soliciting system users’ ideas for better use of the IS. | Communicating an inspiring and appealing vision.  Articulating an appealing vision and expressing high levels of expectation and optimism about the users’ ability. | Admirable behaviours intended to arouse follower emotions and identify with the leader.  Instil pride, faith, and respect in IS users by acting well and leading-by-example; thereby, causing followers to identify with the leader. |
| **Does the Transactional-Transformational Paradigm Transcend Organizational and National Boundaries?**  (Bass, 1997) | Deal with others as individuals; consider their individual needs, abilities, and aspirations; listen attentively; further their development; advise; teach; and coach. | Questions old assumptions, traditions, and beliefs; stimulate in others new perspectives and ways of doing things; and encourage the expression of ideas and reasons. | Articulate an appealing vision of the future, challenge followers with high standards, talk optimistically with enthusiasm, and provide encouragement and meaning for what needs to be done. | Display conviction; emphasize trust; take stands on difficult issues; present their most important values; and emphasize the importance of purpose, commitment, and the ethical consequences of decisions. Such leaders are admired as role models generating pride, loyalty, confidence, and alignment around a shared purpose. |
| **The four I 's of transformational leadership**  (Avolio et al., 1991) | Pay attention to the individual employee and his/her needs rather than treating all followers alike and as having the same needs. Such leaders listen to and share an individual's concerns while simultaneously helping to build the individual's confidence. Perhaps individualised consideration is best represented in the mentoring role.  Makes sure that the employee has the help and resources necessary to achieve current goals.  Each employee has different needs and, that for a specific employee, those needs will change over time partially based on the influence of the leader.  Developing each follower to his/her optimum potential. | Providing ways and reasons for people to change the way they think about technical problems, human relation problems, and even their own personal attitudes and values that have developed over the individual's life span.  Helps people to think about "old" problems in new ways and to use reasoning and evidence rather than unsupported opinion to solve such problems.  Provide reasons why it is in the group's best interest to work as a team to achieve its objectives.  Followers are supported for questioning their own beliefs, assumptions and values, and those of the leader, which may be outdated or inappropriate for solving the problems confronting an organisation  Influences follower's conceptualisation, comprehension, and analysis of problems and the solutions generated. | Set an example of hard work, give "pep" talks, remain optimistic in times of crisis, and search to reduce an employees' duties and workloads by using creative work methods. | Showing respect for others.  Building their confidence and trust in the overall mission.  Convinces his or her followers that by achieving their full potential all concerned will benefit.  Create in followers the ability to lead themselves when necessary and desired.  Idealised influence is, to a large degree, a culmination of the other three I's coupled with a strong emotional attachment to and identification with the leader. Such leaders are emulated by followers and often labelled charismatic. |
| **Leader Behaviors and Their Effects on Followers ’ Trust in Leader , Satisfaction , and Citizenship Behaviors**  (Podsakoff et al., 1990) | The extent to which the leader shows concern for the personal needs and feelings of his or her subordinates. | measures the extent to which  the leader stimulates employees to rethink the way they perform their duties. | developing a vision and getting employees to accept it | being a good example of the values and behaviours that are essential to fulfilling the vision, and getting employees to put the interests of the group or organization above their self-interest. |
| **Transformational Leadership: 1992 and Beyond**  (Bass & Avolio, 1990) | Diagnose followers' needs and attend to them individually. They also delegate, coach, advise, and provide feedback for use in the personal development of followers. They raise the needs and confidence levels of followers to take on greater levels of responsibility. The follower's responsibility does not simply cover their job requirements nor is it geared exclusively to maximising performance. On the contrary, followers are taking greater responsibility for their personal development, which includes such activity as one's job challenges. | Actively encourage a new look at old methods/problems. They foster creativity, and stress a rethinking and re-examination of assumptions underlying problems. They use intuition as well as more formal logic to solve problems.  Develop followers to tackle problems using their own unique and innovative perspectives. | Give pep talks, increase optimism and enthusiasm, and communicate their visions of attainable futures with fluency and confidence.  Provide vision which stimulates the energy to accomplish higher levels of performance and development. | Have a vision and sense of mission.  Gain respect, trust, and confidence.  Acquire strong individual identification from followers.  Obtain the required extra effort from followers to achieve optimal levels of development and performance. |

## *A screenshot of a cell phone Description automatically generated*Timeline of Software Development at SoftwareCorps

## A screenshot of a cell phone Description automatically generatedNetwork Model of Agile Evolution at SoftwareCorps

## Matrix of the Participants’ Views of Agile Principles

| **Agile Principle (#Sources)** | **Participants’ views** | |
| --- | --- | --- |
| **Self-organising teams (10)** | **-** | 1. Senior's “more weight” making argument TL-2 2. Lack of communication = siloed decisions EX-8, TL-6 3. “not interesting” work doesn’t get done D-12 4. “mandated from above” PO-13 5. It's an "agile buzzword" PO-13 |
| **+** | 1. Take “more responsibility”, “deliver” PO-13 2. People who do work decide SM-15 3. Ideas “valued”, “important” TL-3 4. Team solution better than individual PO-9 5. Learn from own mistakes TL-3 6. Good ideas “cross-pollinated” TL-2 |
| **Deliver working software frequently (9)** | **-** | 1. increased overhead delivering small chunks PO-9  2. tendency for code to be "hacked" for speed TL-3 3. Unnecessary on some projects D-12 |
| **+** | 1. Build “incrementally” to ship frequently PO-9, PO-13 2. Releasing early "absolutely invaluable" PO-13 3. Identify “quality problems” early PO-13 4. Respond to change EX-8 5. can “release at any point” TL-2 6. Builds confidence TL-3 7. Business "delivers value regularly" TL-2 8. Satisfaction in getting "it out of the door" TL-3 |
| **Cross-functional teams (9)** | **-** | 1. prioritisation less objective EX-8  2. not economical EX-8, not "cost-effective" to create cross-functional teams TL-6 3. "waste of time" EX-8 |
| **+** | 1. creates alignment across org. PO-9 2. all departments feel “part of the team”, want “success” PO-9 3. "ship things to clients faster" PO-9, EX-8  4. provides an appreciation for other roles and builds trust across departments EX-8, D-12 5. reduces “knowledge silos” TL-5 6. less “throw over the wall” TL-5, TL-2 |
| **Reflect and improve (9)** | **-** | 1. time-consuming, "unnecessary effort" EX-8 2. "intimidating" as a team EX-8 3. "never been a focus" EX-8 4. Learning "wrong things" D-12 |
| **+** | 1. Valuable PO-9 2. "good way to change how you work" PO-9 3. works well EX-8 4. productive, improve stuff TL-2 5. continuous improvement TL-2 6. "embrace change" TL-2 |
| **Satisfy the customer (7)** | **-** | 1. consider "market [vs] individual clients" PO-9  2. Time-consuming & difficult to communicate requirements PO-13, TL-3  3. Clients use the product "very differently" TL-3 |
| **+** | 1. "Understand underlying need" PO-9 2. "important to…get feedback and iterate" PO-9  3. Manage the risk of building "wrong thing" PO-13 |
| **Motivated individuals (7)** | **-** | 1. "a bit fluffy" TL-6 |
| **+** | 1. Feel valued TL-3 2. Requires trust TL-3 |
| **Face to Face conversation (6)** | **-** | None |
| **+** | 1. "Good approach" SM-28 2. "more efficient" EX-8 3. Walls "broken down" D-12 |
| **Promote sustainable development (6)** | **-** | 1. Not "working hard enough" TL-3 |
| **+** | 1. Higher chance of "success" SM-28 |
| **Technical Excellence & Design (6)** | **-** | 1. Not a priority TL-3 2. "Religions" TL-3 |
| **+** | 1. "maintainability" important D-12 |
| **Welcome changing requirements (5)** | **-** | 1. "very disruptive" PO-13 2. "change is costly" PO-13 |
| **+** | 1. Enables "success" PO-13 2. Change "inevitable" PO-13 3. Be "dynamic" TL-2 |
| **Simplicity is essential (5)** | **-** | None |
| **+** | None |
| **Measure progress in working software (1)** | **-** | None |
| **+** | Only done when clients "using it" TL-3 |

1. Dilbert is a comic strip created by Scott Adams which provides satirical commentary on office work. [↑](#footnote-ref-1)
2. Planning Poker is an agile estimating and planning technique that is consensus based [↑](#footnote-ref-2)
3. executive managers (EX), lead developers (TL), developers (D), testers (QA), scrum masters (SM), and product owners (PO) [↑](#footnote-ref-3)
4. 47 interviews were carried out, across 29 participants [↑](#footnote-ref-4)
5. \*'Definition of done' is a Scrum term referring to the agreed-upon tasks which must be completed for the team to consider a piece of work finished, addressing the issue of misinterpretation when individuals say a task is 'done', but the overall piece of functionality is not customer ready. [↑](#footnote-ref-5)
6. *A meritocracy describes a social system where merit or talent determine reward, rather than traits such as race, sex or class* (Alon & Tienda, 2007, p. 189)*. A cooperative model is one where an enterprise is jointly owned by its employees or users and is democratically controlled.* [↑](#footnote-ref-6)
7. *\*The Pareto principle originates from Italian economist Vilfredo Pareto* (Brynjolfsson, Hu, & Simester, 2011) *who found that 20% of the Italian population owned 80% of the land. The principle has gone on to be used liberally in other fields and tends to be applied to anything which conforms to an 80/20 construct, as seen here by Julius.* [↑](#footnote-ref-7)
8. This would be known as ‘I-shaped’ using the concept of letters as metaphors for the range of skills a person has, where an individual has a great depth of knowledge in a single area. [↑](#footnote-ref-8)
9. . A person is considered a ‘key’ dependency when they are essential to the performance of the organisation. [↑](#footnote-ref-9)
10. PDCA originates from process improvement in business and was made popular by W. E. Deming, it is consequently also sometimes referred to as the Deming circle. http://jamme.acmsse.h2.pl/papers\_vol43\_1/43155.pdf [↑](#footnote-ref-10)
11. The term ‘whole method’ refers to adopting all the practices prescribed by a method such as XP or Scrum. In contrast, not using a whole method suggests the users are tailoring the practices for their environment. [↑](#footnote-ref-11)