Software managers' perceptions of Agile software development methods in South Africa

Kalpana Parshotam

A research report submitted to the Faculty of Commerce, Law and Management, University of the Witwatersrand, in partial fulfilment of the requirements for the degree of Master of Business Administration

Johannesburg, 2009

ABSTRACT

Agile methods for software development, typified by their iterative, incremental approach and flexibility to change, became known as such with the founding of the Manifesto for Agile Development in 2001. Agile methods are intended to address the challenges of business requirements and technology changes that continually arise during software development.

This study investigates software managers' perceptions of agile methods for software development in the South African context. The research considered the meaning of agile methods including characteristics, similarities and differences between agile and other software development methods. Software managers' attitudes towards agile methods were examined, covering their views on benefits, shortcomings and challenges of implementing agile methods.

Data was collected through in-depth face to face interviews with fourteen project managers and development team leaders, representing various sizes and types of organisations. Respondents were knowledgeable on the subject of agile methods and had current and/or previous exposure to agile projects.

Important characteristics of agile methods noted were: iterative cycles; using agile methods where appropriate; active business involvement; short planning and delivery cycles and using an appropriate level of documentation.

Comparing agile and other software development methods key differences included: documentation; solution delivery; dealing with changes; planning/delivery cycle and quality assurance/testing. There were considerably fewer references to similarities than differences. Similarities that emerged were: similar concepts but different approach; delivering the end product; team roles/members; project tracking/reporting and producing documentation.

Some of the characteristics of agile methods were also related to the benefits, shortcomings and challenges. Highly rated benefits were: good/better quality; better team spirit/dynamics; meeting client needs/delivering value; improved team performance and early detection of issues/failure.

Shortcomings of agile methods included: limited applicability; cost not necessarily lower and possibly higher; lack of common definition and understanding of agile methods; lower suitability for large projects/teams and lower suitability for distributed/outsourced/off-shore development teams.

Implementation challenges included: selective, customised or combined use of agile methods; resistance to change; the need for education of customers and teams; insufficient business involvement and the need for the right kind of people.

The study revealed that software managers see the meaning of agile software development being strongly grounded in the values and principles of the Agile Manifesto. Agile software development is an iterative approach that breaks work into small packages and delivers a solution incrementally over multiple short cycles, accommodating changes throughout the process. It is appropriate for certain types of situations and environments and should be used accordingly. The characteristics of agile methods are inextricably linked to their distinguishing features. Whilst similarities can be identified at a very high level, agile methods tend to work differently to other methods.

Software managers' attitude towards agile software development is largely positive, improved quality being viewed as the most significant benefit. These benefits arise from the way agile projects work and specific agile practices such as pair programming, continuous integration and test-driven development.

While the general sentiment is positive, agile methods have inherent limitations which mainly centre on their limited applicability with regard to certain types of situations or projects. Agile methods are not seen to reduce project costs although quality and time are generally positively impacted.

Agile methods being new and different face resistance to change during implementation. Education is needed to overcome resistance and lack of understanding of agile methods. Software managers should appreciate that agile methods should be implemented with situation-dependent adaptation as appropriate.

DECLARATION

I, Kalpana Parshotam, declare that this research report is my own work except as indicated in the references and acknowledgements. It is submitted in partial fulfilment of the requirements for the degree of Master of Business Administration in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in this or any other university.

Kalpana Parshotam

Signed at ……

On the …….. day of …… 2009

DEDICATION

To all software practitioners (agile proponents and sceptics alike) who are passionate about their field, thriving on the challenges of software development and delivering useful systems.

ACKNOWLEDGEMENTS

My sincere thanks to:

- My parents for their constant encouragement
- My husband for his patience and support
- My supervisor Antony Soicher for his advice and support
- My previous and present employers The IQ Business Group and First National Bank respectively for time off and access to resources
- Willy-Peter Schwaub for the copy of his book ‘Software Engineers on their way to Pluto’
- Jayvant Parshotam, Liz Hazell and Stefané Haupt for proof-reading and review
- All research participants for providing a wealth of information that has increased my knowledge and insight even beyond the scope of this specific research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Purpose of the study

The purpose of this study is to investigate software managers' perceptions of agile methods for software development in South Africa. The study attempts to understand the current mindset regarding agile software development methods in South Africa, within the practical setting of software development organisations, i.e. companies or departments within companies that undertake software development. The focus is on business management rather than technical issues; hence the study investigates perceptions from the perspective of those who manage software development teams, i.e. project managers and development team leaders. The study is exploratory in nature, aiming to obtain a deeper understanding of the agile phenomenon in the South African context.

1.2 Context of the study

Software development today is taking place in an environment of high-speed and unpredictable change on both the business and technology fronts. So-called traditional software development methods (Highsmith and Cockburn 2001; Glass 2001; Boehm 2002; Baskerville, Ramesh, Levine, Pries-Heje and Slaughter 2003; Boehm and Turner 2005; Nerur et al 2005; Haynes and Friedenberg 2006) are also referred to as disciplined (Cusumano and Yoffie 1999; DeMarco and Boehm 2002; Baskerville et al 2003; Beck and Boehm 2003), plan-based (Boehm 2002; Cockburn 2002B; Williams 2004; Ceschi, Sillitti, Succi and De Panfilis 2005; Fowler 2005; Nerur, Mahapatra and Mangalaraj 2005; Haynes and Friedenberg 2006), Tayloristic (Fowler 2005; Melnik and Maurer 2005) or engineering-based methods (Fowler 2005; Nerur et al 2005). The waterfall method typifies the traditional approach and is generally implemented as a sequential development lifecycle (Cusumano and Yoffie 1999; Williams 2004; Nerur et al 2005; Haynes and Friedenberg 2006), although Winston Royce the originator of this method recommended a two-pass approach through the development cycle (Larman and Basili 2003). These

methods focus on following a tightly defined software development process, where system requirements are elucidated and documented upfront, and the project is planned and controlled to ensure delivery of these requirements.

However, these approaches face shortcomings, particularly in the area of requirements management. Potential users of the software may not always know what they want (Beck 1999; Theunissen 2003), often don't say what they want, have difficulty stating their requirements (Theunissen 2003; Grossman, Bergin, Leip, Merritt and Gotel 2004), contradict themselves and change their minds (Beck 1999). People generally expect that requirements ought to be changeable (Fowler 2005). Business models and processes are evolving in response to competition resulting in requirements change (Theunissen 2003). Requirements for systems also change in order for systems to be more closely aligned with business goals (Theunissen 2003). Requirements are often presented at a high level making it difficult and time-consuming to define detailed specifications upfront (Lindvall, Muthig, Dagnino, Wallin, Stupperich, Kiefer et al 2004). Requirements and project plans thus become out of date relatively quickly (Williams and Cockburn 2003).

Software development organisations are also feeling pressure for increased productivity at lower costs whilst still maintaining or improving quality (Lindvall et al 2004).

While traditional methods focus on containing and managing change during the lifecycle of a project, the agile methods embrace change as being inevitable, and suggest a process that can create and respond rapidly to change (Cockburn and Highsmith 2001). Proponents such as Highsmith and Cockburn (2001) see recent agile software development methods addressing the challenge of constant change by reducing the cost of responding to change, whilst still promoting quality. The technical and managerial processes in agile approaches are intended to continually adapt and adjust to changes resulting from experiences during development, changes in requirements and changes in the development environment (Turk, France and Rumpe 2002; 2005).

In February 2001, practitioners of different agile methods reached agreement on the common underlying elements of their methods. In The Manifesto for Agile Development (Beck, Beedle, Cockburn, Cunningham, Fowler and Grenning et al 2001) the common values and principles were documented, as stated in section 2.3.1.

The agile approach encompasses a number of agile methods that have evolved over time in different countries. Each of these methods prescribes its own specific practices. Some of these methods are detailed in section 2.2.3.

While many of the agile methods have been in practice for a number of years, as acceptable development approaches they have only recently come to the fore in the past 5 to 7 years. Although there is literature expounding the rationale, benefits, practices and processes of agile methods, it remains a question as to what practitioners in South Africa actually understand and think about agile software development, irrespective of whether or not opinions are informed by practical experience.

1.3 Problem statement

1.3.1 Main problem

The aim of this research is to investigate the perceptions of agile software development methods from the perspective of software management practitioners in software development organisations in South Africa.

1.3.2 Sub-problems

The first sub-problem is to investigate what software management practitioners in South Africa understand by agile software development.

The second sub-problem is to investigate what are the general attitudes of software management practitioners towards agile software development methods in South Africa.

1.4 Significance of the study

This study fills a gap in that it aids uncovering the level of awareness and attitudes around agile methods amongst the software development management community, by looking at the perceptions of these approaches in South Africa. Ultimately it is software managers' understanding and opinions of agile software development that influences the extent to which agile methods are adopted, and whether these methods become mainstream or a passing fad. Irrespective of what the literature might suggest or what software developers might think, it is the managers accountable and responsible for the delivery of projects that decide how projects are run.

This study can provide guidance to members of the software development community. Specifically software managers, i.e. those occupying development team leader and project management roles may find practical application from the factors suggested by the research findings. The research findings may provide insight into the management perspective of software development projects, to developers and other development team members.

This study may also be of interest to business people as customers of software. They may feel more amenable to participation in agile projects based on software managers viewing agile methods positively.

Educators and trainers in the field of software development and software project management may also find this study of interest, as it may highlight areas of emphasis for education and training of undergraduates and practitioners.

Finally this study may be of interest to academics as thus far research in South Africa in the field of agile methods has been limited. South African research to date includes: case study research on Extreme Programming in the telecommunications industry (Theunissen 2003); development of guidelines for adapting agile methods to conform to international standards (Theunissen, Kourie and Watson 2003); comparison of open source development with agile development (Theunissen, Boake and Kourie 2005); case study research on organisations using rapid development methodologies (Gerber, Van der Merwe

and Alberts 2007); development of a selection tool for applying agile practices to a project (Mnkandla and Dwolatzky 2007) and empirical research into stakeholder satisfaction on agile projects (Ferreira and Cohen 2008). This research may shed further light on the agile phenomenon in South Africa providing stimulus for further research into this area.

1.5 Delimitations of the study

The study is delimited to software development organisations based in South Africa for reasons of access, time and cost for this research.

The study is delimited to exploring the problem from the perspective of persons managing software development projects (i.e. project managers and software development team leaders) within software development organisations. The perspectives of other software development team members internal to these organisations (e.g. developers) and customers external to these organisations have not been included.

The study does not attempt to directly observe the execution of software development projects, nor examine the artefacts of the projects. Any assessment as to whether agile methods and specific practices are being applied and the effectiveness of such methods and practices is made by the participants in the study and not by the researcher.

The research does not attempt to empirically measure or analyse software development project processes, productivity or outcomes. The research looks at perceptions only, and does not seek to cross-check these perceptions against actual experience.

1.6 Assumptions

This research is based on the following assumptions:

a. The participants in the study were involved in software development projects in a direct capacity such that they were able to relate the concept of agile software development meaningfully to a practical work setting. If the participants have not responded based on their experience in software development projects, the validity of the study will be impacted.

b. Information was conveyed honestly and truthfully by the respondents. False data will detrimentally affect on the study's results.

c. The research instrument used, i.e. face to face interviews, is sufficient to generate meaningful data without loss of reliability and validity.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter reviews the base of literature covering the subject of agile methods in order to provide a background and basis against which research findings can be interpreted. The background discussion begins by integrating various definitions of agile methods from the literature into a working definition. The history of agile methods is outlined and then an overview of some of the specific agile methods is presented.

The review then explores the meaning of agile software development, in terms of its values, principles, characteristics and situational application. Agile methods are then contrasted with other software development methods and some of the differences and similarities are presented. Thereafter the benefits and shortcomings of agile software development methods are discussed. Finally the challenges of implementing agile methods are examined.

2.2 Definition and background

2.2.1 Definition

In surveying the literature one realises that the different authors do not present a common, succinct definition of agile development. Conboy and Fitzgerald (2004) suggest that there is not a universally accepted definition of agile methods for software development. In describing agile methods many authors refer to the Agile Manifesto values (Glass 2001; Highsmith and Cockburn 2001; Abrahamsson, Salo, Ronkainen, and Warsta 2002; Boehm 2002; Cohn and Ford 2003; Williams and Cockburn 2003; Lindstrom and Jeffries 2004; Chow and Cao 2008; Dyba and Dingsøyr 2008) and principles (Glass 2001; Turk et al 2002; Baskerville et al 2003; Conboy and Fitzgerald 2004; Lindstrom and Jeffries 2004; Williams 2004; Turk et al 2005), which are also presented in this paper in section 2.3.1.

Drawing from various authors the definition of agile software development methods encompasses the following concepts: Agile methods adapt proactively and reactively to change (Cockburn and Highsmith 2001; Kruchten 2001; Abrahamsson et al 2002; Highsmith 2002; Conboy and Fitzgerald 2004) in response to changes in the environment (Kruchten 2001; Cockburn and Highsmith 2001; Cockburn 2002B; Erickson, Lyytinen and Siau 2005) and changing stakeholder needs (Erickson et al 2005; Melnik and Maurer 2005; Ambler 2007). This adaptability comes from using a flexible process (Kruchten 2001; Highsmith 2002; Chow and Cao 2008) that is lightweight (Boehm and Turner 2005; Erickson et al 2005; Ambler 2007) and straightforward (Abrahamsson et al 2002).

The delivery approach in agile methods is iterative and incremental, using short delivery cycles and small releases (Abrahamsson et al 2002; Boehm and Turner 2005; Ambler 2007). The approach is people-centred (Cockburn and Highsmith 2001; Melnik and Maurer 2005) and relies on a high degree of collaboration (Abrahamsson et al 2002; Melnik and Maurer 2005; Ambler 2007) and communication amongst the development team and with customers (Abrahamsson et al 2002; Melnik and Maurer 2005). Teams themselves determine how they work, i.e. they are self-organising (Boehm and Turner 2005; Ambler 2007).

Summarising the concepts above agile software development methods can be defined as a set of methods that are adaptive, flexible, lightweight and people-centred, delivering solutions with self-organising, collaborative teams using an iterative, incremental approach to continually respond to changing environments and stakeholder needs.

2.2.2 History of Agile Software Development Methods

There is an extensive body of knowledge describing the different agile development methods. As an overview, Figure 1 below illustrates how the different agile software development methods have evolved. The methods that have contributed towards The Agile Manifesto are shown with a dashed line in the diagram.

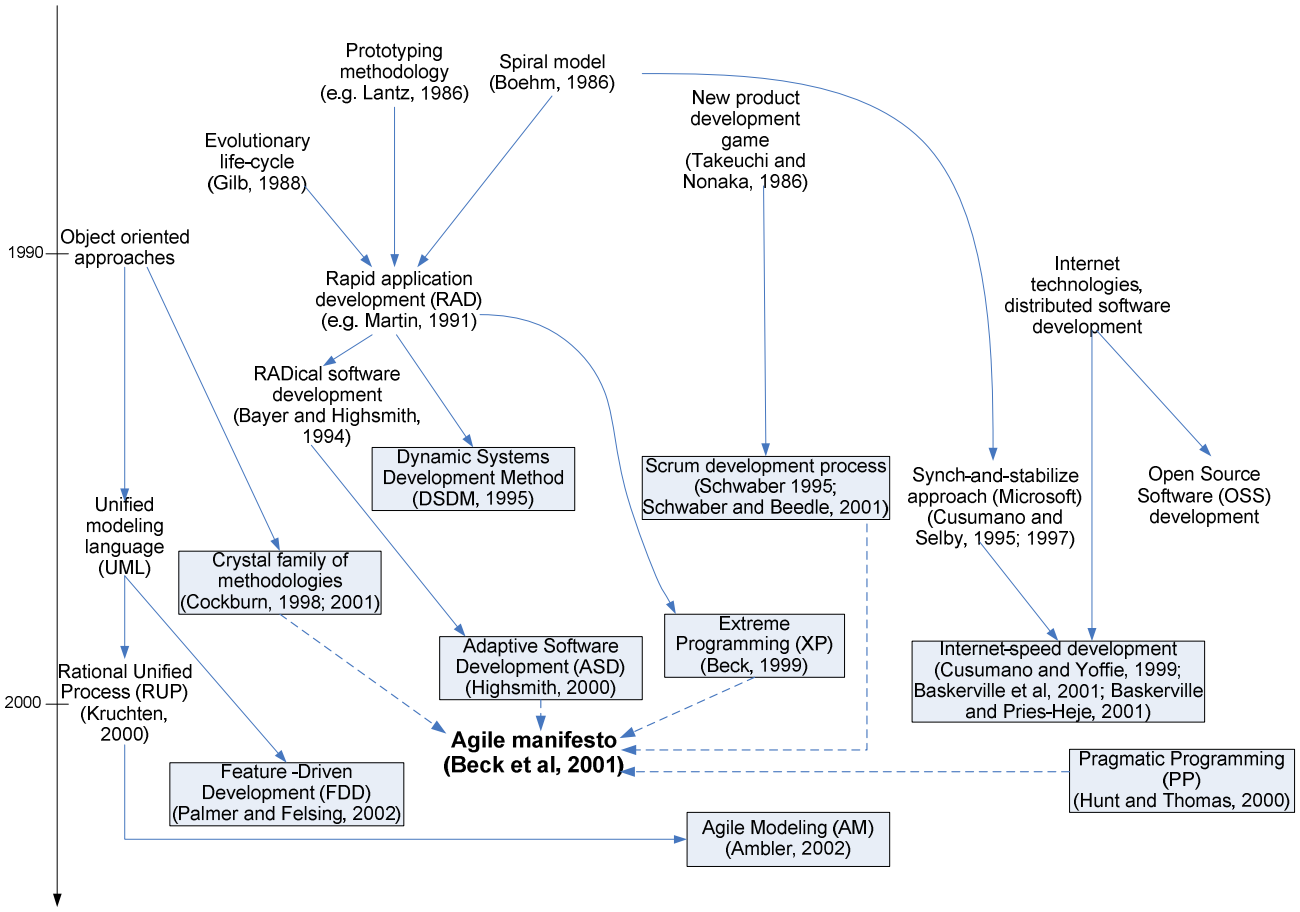

Figure 1: Evolutionary map of agile methods

Source: Adapted from Abrahamsson, Warsta, Siponen and Ronkainen (2003:246)

Larman and Basili (2003) trace how iterative and incremental development (IID) has evolved and has been incorporated into the various agile methods. Whilst agile methods are relatively new, IID has been practiced from the 1930's. IID was used in a number of projects through the following decades in various guises.

Barry Boehm's spiral model, a response to the waterfall model (Beck 1999) was published in 1986 and is regarded as a landmark publication on the subject of IID. In 1988, Tom Gilb, building on his earlier work in the 1970's, published the first book containing substantial content on IID (Larman and Basili 2003).

In the 1990's public awareness of IID in software development grew and the United States Department of Defence (DoD) began advocating the use of iterative development (Larman and Basili 2003). In the commercial space Jeff Sutherland, Ken Schwaber and Mike Beedle began applying a highly iterative development methodology, later to become known as Scrum (Larman and Basili 2003; Fowler 2005).

In the late 1980's and early 1990's Microsoft began refining a flexible style of software development, referred to as "synchronize and stabilize", as an alternative to sequential product development, (Cusumano and Yoffie 1999). This approach described how Microsoft enabled multiple small teams and individual programmers to work together as a single large team to build systems relatively quickly (Cusumano and Selby 1997).

Cusumano and Yoffie (1999) in their study of "Software Development on Internet Time" researched how Microsoft and Netscape were using the "synchronize and stabilize" method for Internet development. In 2000 and 2001 Baskerville and Pries-Heje (2002) investigated how "Internet Speed" was influencing software development and conducted research in the United States and Denmark, to identify factors that characterise Internet Speed development.

The Microsoft Solutions Framework (MSF) process model has its origins in the model used by Microsoft for development of its own applications. MSF, incorporating the concepts synchronisation and stabilisation (Microsoft Corporation 2002), was originally introduced in 1994 and has undergone a number of iterations (Microsoft Corporation 2003; Turner 2006). MSF Version 3.0 introduced in 2002 was stated to be compatible with agile practices but also suitable for projects requiring a higher level of structure (Microsoft 2003). MSF Version 4.0 went a step further in the agile direction, offering two application development approaches namely MSF for Agile Software Development and MSF for CMMi (Capability Maturity Model Integration) Process Improvement (Young 2005; Turner 2006).

In 1994, Rapid Application Development (RAD) practitioners met to define a standardised iterative process that would support RAD. This process definition

formed the basis for the Dynamic Systems Development Method (DSDM) (Larman and Basili 2003).

In the mid-1990's another IID method, the Rational Unified Process was developed by Philippe Kruchten, Ivar Jacobsen and others at Rational Corporation (Abrahamsson et al 2002; Larman and Basili 2003). .

Kent Beck together with Ward Cunningham worked on developing an adaptive and people-oriented software development approach through the late 1980's and early 1990's (Fowler 2005). In 1996, Beck developed the full set of Extreme Programming (XP) practices during the Chrysler C3 Payroll project (Larman and Basili 2003; Fowler 2005).

In 1997, Peter Coad and Jeff de Luca turned a failing project for a large banking application around by applying IID (Abrahamsson et al 2002; Highsmith 2002; Larman and Basili 2003). De Luca, incorporating Coad's ideas then created Feature Driven Development (FDD) (Larman and Basili 2003).

Pragmatic Programming is not a method per se but a group of programming best practices published by Andrew Hunt and David Thomas, who were both also participants in the Agile Manifesto (Abrahamsson et al 2002).

James Highsmith, one of the co-authors of RADical Software Development built on his research into iterative development to create Adaptive Software Development (ASD) in 2000 (Abrahamsson et al 2002; Highsmith 2002).

Alistair Cockburn developed the Crystal family of development methods as a set of approaches tailored for teams of different sizes (Fowler 2005) in 2001. The different methods were labelled with colours with the most agile version being Crystal Clear, followed by Crystal Yellow, Crystal Orange, and Crystal Red (Williams 2004).

Acceptance of IID has grown over the years, with both the Standish Report (in 1998) and the DoD (in 2000) recommending the use of IID (Larman and Basili 2003).

In February 2001, software process experts interested in promoting a modern, simple IID approach met in Utah in the United States. With representation from various agile camps, the meeting aimed at achieving common ground. The outcome was the Agile Manifesto and the catch phrase “agile methods” that encompasses a number of methods, all of which apply IID (Larman and Basili 2003).

2.2.3 Specific Agile Software Development Methods

There are a host of software development methods that are regarded as agile. Methods that are more commonly regarded as being agile with their authors also participants in the Agile Manifesto (Highsmith 2002) include: Extreme Programming (XP), Scrum, Adaptive Software Development (ASD), Dynamic Systems Development Method (DSDM), the Crystal Family of Methodologies, and Feature Driven Development (FDD) (Abrahamsson et al 2002; Boehm 2002; Highsmith 2002; Abrahamsson, Warsta, Siponen and Ronkainen 2003; Williams 2004; Chow and Cao 2008).

Other methods also regarded as agile include: Lean Development (LD) (Boehm 2002; Highsmith 2002; Abrahamsson et al 2003; Williams 2004; Chow and Cao 2008), Agile Modelling (AM) (Boehm 2002; Abrahamsson et al 2003), Internet Speed Development (ISD), “Synch-and-Stabilize” from Microsoft, Pragmatic Programming (PP) (Abrahamsson et al 2003), Agile MSF (Ambler 2006), Rational Unified Process (RUP), Open Source Software Development (Abrahamsson et al 2002), Agile Unified Process (AUP) and Open UP (VersionOne Inc. 2007; VersionOne Inc. 2008).

While it is beyond the scope of this paper to provide extensive treatment on the details of all of these methods, a brief overview is presented for three of the commonly used agile methods, i.e. XP, Scrum and Agile MSF. These descriptions also serve to illustrate the extension of the broader agile principles into some of these methods’ defined practices.

XP is regarded as one of the most well-known (Theunissen et al 2003), widely used (Theunissen et al 2003; Newkirk 2003; Lindstrom and Jeffries 2004) and

prescriptive (Newkirk 2003) agile methods. The popularity of XP and Scrum is evidenced by their ranking in the top three in a number of surveys on the use of agile methods (Ambler 2006; VersionOne Inc. 2007; VersionOne Inc. 2008).

a. Extreme Programming (XP)

According to Beck (1999), XP focuses on reducing the cost of change through short development cycles that incorporate planning, analysing and designing continuously throughout software development, as shown in Figure 2. XP is seen to be clearly applicable to environments with in-house or outsourced development, for small to medium sized systems where requirements are vague and are likely to change.

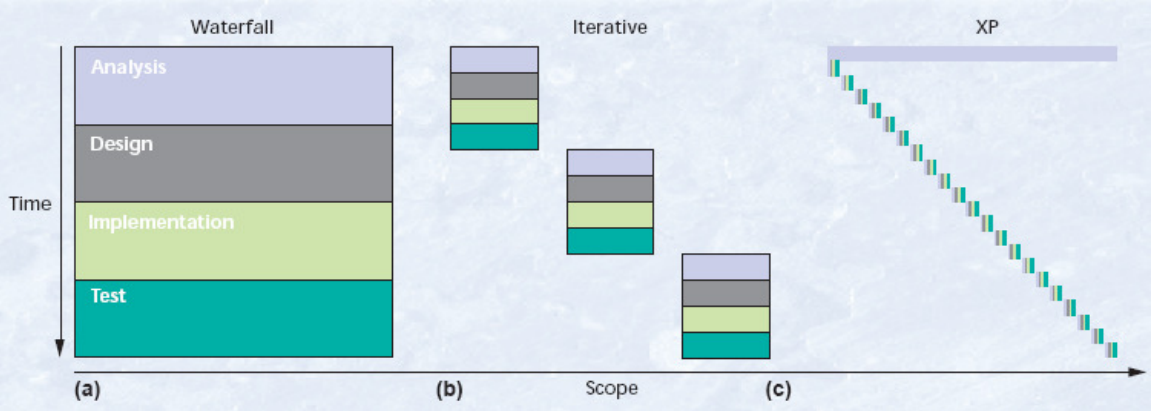

Figure 2: The evolution of waterfall to iterative to XP model

Source: Beck (1999:70)

XP uses user stories as the basis for the customer requirements for development. A story must be "business-oriented, testable and estimable" (Beck 1999:71). The customer chooses the smallest logical set of valuable stories to form part of a release (Beck 1999), a release being a supported product to be delivered to customers (Williams 2004). The customer selects the most valuable stories from the release to form part of an iteration (Beck 1999), which serves as an internal deadline for the team and a checkpoint for the customer (Williams 2004). Within an iteration the team breaks up stories into tasks; developers sign up to work on these stories and provide estimates on them (Beck 1999).

The XP process is built around programmer-centric, technical practices designed to work together (Beck 1999; Abrahamsson et al 2002; Newkirk 2003; Lindstrom and Jeffries 2004; Williams 2004). In the "planning game" customers participate to decide on the scope and timing of releases and iterations. The system is provided to the customer with increasing functionality over small, frequent releases. A system metaphor is developed by the team as a shared understanding of the high-level vision. A simple design is adopted as in the simplest solution to deliver current customer-required functionality with no duplication or additions relating to possible future changes. Testing incorporates acceptance tests written by the customer and unit tests written by programmers. Refactoring caters for continually evolving and improving the existing design without changing functionality (continuous design improvement). With pair programming, all code is written by two programmers working together. All working code is integrated into the current system several times per day, i.e. continuous integration. Collective code ownership is accepted once code is checked in, and code can then be changed by any team member. An on-site customer sits with the team full-time. A 40 hour week, i.e. no excessive, sustained overtime is encouraged. The team works in a single room in an open workspace. The team uses consistent coding standards. After each iteration the team reflects on what went well and what should be improved ("Retrospective"). Self-directed teams allow decision-making to take place closest to the detail. Beck (1999) notes that the aforementioned are "just rules" and the team can change them provided they agree on how to assess the effects of the changes.

Guiding the XP practices are four values (Newkirk 2003; Lindstrom and Jeffries 2004; Turk et al 2005): communication between all team members including customers, writing software for today (simplicity), early and frequent feedback, and asking participants to play to win (courage).

b. Scrum

Scrum is regarded as a process for building software in complex environments, using small teams of ten or less people (Rising and Janoff 2000) or for projects with rapidly changing or highly evolving requirements such as web development

or new market product development (Cohn and Ford 2003). An overview of the Scrum process is shown in Figure 3.

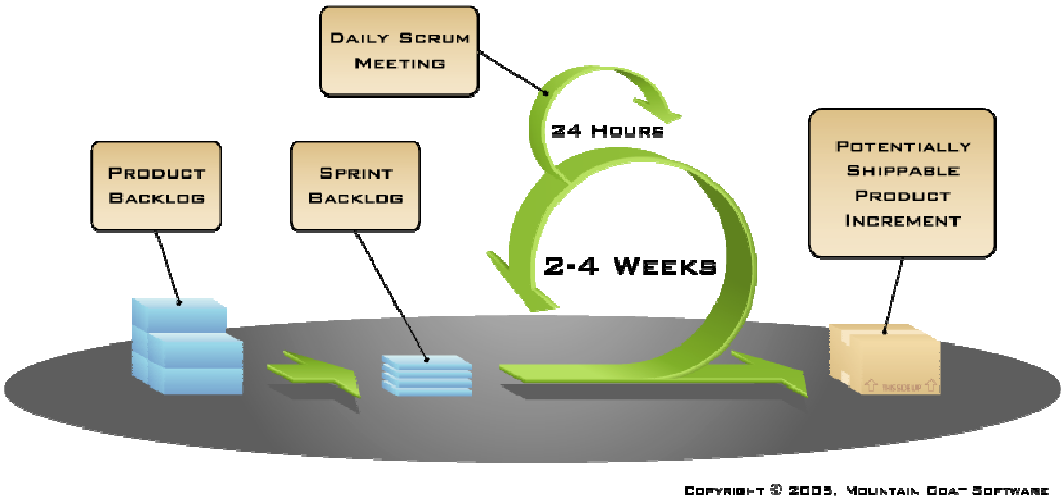

Figure 3: Scrum development process

Source: Cohn (2005) (with permission from the author)

The Scrum process delivers software in 30 day increments, known as Sprints (Rising and Janoff 2000; Abrahamsson et al 2002; Cohn and Ford 2003). The sprint is time boxed and aims to produce a visible, usable, potentially shippable product that implements one or more valuable functions in the system (Rising and Janoff 2000; Cohn and Ford 2003). The sprint begins with Sprint Planning, where the customer creates and prioritises the Product Backlog. The Product Backlog is an evolving prioritised list of business and technical functions to be developed as well as defects to be addressed in the release.

Team members hold a daily 15 minute Scrum meeting, where they feed back on progress since the last Scrum, any obstacles faced, and actions to be undertaken up to the next Scrum. Code is integrated and regression tested on a daily basis (Rising and Janoff 2000; Williams 2004; Schatz and Abdelshafi 2005). A burndown chart is used to track the remaining work on a sprint; the burndown chart graphs the hours of work remaining day by day and is displayed prominently for the team (Williams 2004). At the end of each sprint, a Sprint Review takes place where new functionality is demonstrated to the client,

leading into the next cycle with the next Sprint Planning (Rising and Janoff 2000; Williams 2004; Schatz and Abdelshafi 2005).

c. MSF for Agile Software Development

MSF for Agile provides a complementary set of practices for agile software development (Miller 2005). MSF for Agile is appropriate for projects with short lifecycles using results-focused teams that are able to work without extensive documentation (Young 2005). MSF for Agile is part of MSF version 4.0. MSF 4.0 describes five tracks for project governance, which correspond largely with previously defined MSF phases, i.e. Envision, Plan, Develop, Stabilise and Deploy. Each track covers a set of work streams and activities. The tracks need not be sequential and can run in parallel with communication and feedback between the tracks (Young 2005).

MSF 4.0 uses five cycles that represent iterations that are distinct from the governance tracks. These cycles are: Check-In, Daily Build, Accepted Build, Iteration and Project (Young 2005).

In keeping with the iterative concept MSF 4.0 defines five governance checkpoints that are reviewed continuously at every iteration rather than necessarily at project phase boundaries. The checkpoints ask questions relating to: achievement of acceptance criteria, progress achieved, quality levels, project and technical risk mitigation and deployment readiness of the solution (Young 2005).

In MSF for Agile functional requirements are represented by “scenarios”. Development takes place in short time-boxed iterations where work is scheduled from the scenario list and a list of “quality of service” requirements (e.g. performance, platform, security). The business analyst details each of the scenarios and these are divided by the developers into smaller tasks for which costs are provided. Planning takes place in a staggered manner such that the business analyst and project manager may be analysing requirements and planning a future iteration while the development for the current iteration is taking place. Customers are represented using “personas”, which are fictitious people that represent groups of users and describe their goals, purpose,

knowledge, usage patterns and concerns. MSF uses the concept of "shadowing" where the shadow is the architecture for the work to be developed in a specific iteration. Testing is context-based, i.e. the testing approach and the test thresholds are developed by the project team specifically for a project (Miller 2005).

2.3 Meaning of agile software development

The meaning of agile methods is underpinned by the values and principles set out in the Agile Manifesto. These values and principles manifest in specific characteristics of agile methods, which are reflected to a greater or lesser extent in the different agile methods. The characteristics of agile methods in turn lend themselves to a more natural applicability to certain project and environmental situations. Agile methods by their nature tend to be very different to traditional software development methods but they do reflect some similarities as well.

2.3.1 Describing Agile Development

a. Values

The Manifesto for Agile Development (Beck et al 2001) states the common values underlying agile approaches:

> "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."

The values stated above reflect the conception of software development as a flexible, people-centred and creative process for which the ultimate measurement is in the production of functioning software.

a. Principles

The underlying principles that agile proponents subscribe to are stated in the Manifesto for Agile Development (Beck et al 2001). These principles closely reflect the values outlined above, and form the basis of practices intended to manage change.

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

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

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

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

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

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

7. “Working software is the primary measure of progress.”

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

9. “Continuous attention to technical excellence and good design enhances agility.”

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

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

12. “At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.”

These principles display the importance of the development team (Principles 5 and 11), collaboration and communication (Principles 4 and 6), quality (Principle 9), regular delivery of working software (Principles 1, 3, 7, 10), sustainability (Principle 8), and learning (Principle 12), in relation to dealing with change (Principle 2).

b. Characteristics of agile software development

The values and principles of agile software development manifest in specific characteristics of agile software development.

i. Iterative incremental development cycles

Agile projects are delivered in short, iterative, feature-driven, time-boxed development cycles or sub-projects (Highsmith and Cockburn 2001; Highsmith 2002; Cockburn 2002A; Boehm and Turner 2005; Miller 2005; Nerur et al 2005; Turk et al 2005).

Agile methods advocate an iterative approach, i.e. a project is completed over several cycles (Boehm and Turner 2005). Each cycle involves planning, development, integration, testing and delivery (Nerur et al 2005).

The duration of a sub-project or iteration is short. Iteration lengths vary with most authors indicating cycles of one to eight weeks (Highsmith and Cockburn 2001; Cockburn 2002A; Williams 2004; Fowler 2005; Miller 2005), but sometimes up to twelve weeks (Cockburn 2002A). New releases could be produced as frequently as hourly or daily in some cases (Abrahamsson et al

2002; Miller 2005), but are more usually produced bi-monthly or monthly (Abrahamsson et al 2002).

The approach is also incremental (Cockburn and Highsmith 2001; Boehm and Turner 2005), i.e. the entire solution is not delivered at once (Boehm and Turner 2005).

ii. Feature planning

Agile methods focus on planning features not tasks, as features are what customers are able to identify with. Different methods incorporate feature planning in various ways: XP makes use of story cards, Scrum uses a "backlog" (Highsmith and Cockburn 2001) and ASD focuses on the delivery of application features per iteration (Highsmith 2002).

A pre-specified iteration length is used as a time-box, and the scope is selected to fill the iteration based on prioritised functionality (Williams 2004; Boehm and Turner 2005).

Iterations are focused on definitive, customer-relevant chunks (Highsmith 2002). The customer and the development team decide jointly which features are to be implemented in a cycle (Nerur et al 2005).

iii. Early and regular delivery of working code

The short iterations provide early and regular delivery to the customer. Each sub-project or cycle should deliver a useful section of the system (Cockburn 2002A; Nerur et al 2005) that is integrated across the team (Williams 2004), tested and working (Abrahamsson et al 2002) and represents an emergent subset of the final solution (Williams 2004). Frequent short releases ensure that highest priority functions are implemented, deliver customer value rapidly and allow requirements to emerge quicker (Boehm and Turner 2005). The focus on delivering value early mitigates the risk of non-fulfilment of the contract (Abrahamsson et al 2002).

Of importance is that working code is produced (Cockburn and Highsmith 2001; Highsmith and Cockburn 2001; Highsmith 2002; Nerur et al 2005). Frequent

delivery of working code provides visibility of the project in a way that a customer can understand (i.e. an executable product) rather than in the form of documentation that a customer may not easily relate to (Turk et al 2005). Highsmith and Cockburn (2001) emphasise that the working code demonstrates to developers and sponsors something that is real and could be shipped, modified or scrapped as opposed to promises on what they might have. Working software also allows one to measure the rate at which output is produced and provides for rapid feedback (Cockburn and Highsmith 2001).

iv. Feedback and change

The short iterations promoted by agile methods support the incremental approach. Short iterations provide for timely customer feedback (Karlström and Runeson 2005; Turk et al 2005) ensuring that the end product meets customers' needs (Turk et al 2005). Customer feedback based on observation of the evolving product in earlier iterations can be incorporated into future iterations (Williams 2004).

Early and regular deliveries to customers provide opportunities for customers to assess project progress, clarify and refine requirements (Turk et al 2005) and provide feedback on the system in operation and identify where initial thoughts may have been mistaken (Cockburn 2002A). The team can also adjust to any new information and incorporate refinements or new requirements (Highsmith and Cockburn 2001; Cockburn 2002A; Williams 2004).

Agile methods facilitate dynamic prioritisation (Highsmith and Cockburn 2001). At the end of a cycle a customer can reprioritise features or re-plan for the next cycle (Highsmith and Cockburn 2001; Cockburn 2002A; Highsmith 2002) based on the current reality (Highsmith 2002). A customer may discard originally planned features and/or add new features (Highsmith and Cockburn 2001).

Agile methods embrace change (Highsmith and Cockburn 2001; Abrahamsson et al 2002; Highsmith 2002; Boehm and Turner 2005: Nerur et al 2005). Accommodating changes allows for greater creativity and more rapid value to be delivered to the customer (Boehm and Turner 2005). In order to deal with changes the development team and customer must be competent and well-

informed to be able to take into account potential changes that might occur. They must be willing to make changes and the contracts should be created with tools that support such changes being made (Abrahamsson et al 2002).

v. Customer involvement and collaboration

Customers or users of the solution are actively involved in the agile process (Cockburn and Highsmith 2001; Abrahamsson et al 2002; Highsmith 2002; Boehm and Turner 2005; Nerur et al 2005; Turk et al 2005). Users are involved to provide, prioritise and confirm their requirements (Boehm and Turner 2005). Users are able to provide feedback on items that developers may have misunderstood and identify items that might not be working as well in practice as initially thought (Cockburn and Highsmith 2001).

A high level of customer collaboration and interaction is required with the sponsor, customer, user and developer on the same team (Cockburn and Highsmith 2001; Highsmith and Cockburn 2001; Abrahamsson et al 2002; Highsmith 2002; Fowler 2005; Nerur et al 2005). Whilst contracts are important, collaboration is still required (Highsmith and Cockburn 2001; Abrahamsson et al 2002). Close collaboration allows for better mutual understanding and improves the chances of project success (Turk et al 2005). Working together allows the combined group to be able to change direction quickly and produce more appropriate and less expensive solutions (Highsmith and Cockburn 2001).

vi. Lightweight process

Agile methods are lightweight (Boehm and Turner 2005) and advocate the concept of “barely sufficient” methodology that balances flexibility and structure. Bare sufficiency streamlines the process thereby lowering costs, and also incorporates the perspective that creativity and innovation tend to occur in more unstructured environments (Highsmith 2002).

Documentation is minimised (Cockburn and Highsmith 2001; Cockburn 2002A; Nerur et al 2005) with a greater emphasis on face to face communication, the use of whiteboards (Cockburn and Highsmith 2001; Turk et al 2005) and informal communications including pair programming and daily stand-up

meetings (Cockburn 2002A). With a lower emphasis on documentation knowledge tends to be more tacit (Boehm and Turner 2005; Nerur et al 2005), i.e. knowledge resides more in participants' heads as opposed to being explicit, i.e. in documents (Boehm and Turner 2005).

vii. Team proximity and interaction

Close team proximity is also a feature of agile methods. Teams should be physically closer or collocated (Highsmith and Cockburn 2001; Cockburn and Highsmith 2001; Abrahamsson et al 2002; Cockburn 2002A) as suggested by Scrum, ASD (Highsmith and Cockburn 2001), Crystal and XP (Highsmith and Cockburn 2001; Cockburn 2002A). Close team interaction, communication and relationships are also emphasised (Highsmith and Cockburn 2001; Abrahamsson et al 2002; Cockburn 2002A). Developers tend to work in small teams (Nerur et al 2005).

The high level of interaction and face to face communication improves effectiveness in a number of ways: ideas can be exchanged faster than through documents; designers can produce better designs working with each other than alone; difficulties can be overcome, priorities adjusted and alternative paths explored through discussions between developers, customers and sponsors; and information can be shared easily and processes quickly changed where individuals communicate with each other (Highsmith and Cockburn 2001).

viii. People and roles

Agile methods recognise people to be a highly influential factor to a greater extent than processes or tools (Williams 2004) and that people working well together can be highly effective (Highsmith and Cockburn 2001). There is an emphasis on improving team morale or spirit (Cockburn and Highsmith 2001; Abrahamsson et al 2002; Cockburn 2002A). Agile organisations also focus on developing individual team members' skills (Highsmith 2002).

Team members can rotate roles rather than have specialised roles, and have more discretion and decision-making power, creating more diversity within teams with knowledge spread across multiple team members as opposed to

individuals (Nerur et al 2005). Teams have increased capability to self-organise (Highsmith 2002; Boehm and Turner 2005; Nerur et al 2005), i.e. decide within the team the best way to carry out work (Boehm and Turner 2005), make technical decisions (Fowler 2005) and respond to situations (Nerur et al 2005). Decisions are taken collectively (Nerur et al 2005) and processes and work structures emerge during the project as opposed to being predetermined (Boehm and Turner 2005). The project manager role focuses on facilitation and coordination using a collaborative management style (Nerur et al 2005).

ix. Process review

Agile methods incorporate a period of reflection at the end of each cycle (Cockburn 2002A; Boehm and Turner 2005; Fowler 2005; Nerur et al 2005). These reviews assess the effectiveness of work, methods utilised and estimates used (Boehm and Turner 2005). Reviews support team learning (Boehm and Turner 2005; Nerur et al 2005) and create opportunities for change (Nerur et al 2005), remedying mistakes, trying out ways to increase effectiveness (Cockburn 2002A) and as input for estimating future iterations (Boehm and Turner 2005). Each delivery also builds confidence for sponsors, customers and developers in the way that they are working (Cockburn 2002A).

x. Agile practices

Agile approaches suggest specific practices that cater for collaboration, constant feedback and change (Highsmith and Cockburn 2001; Highsmith 2002; Nerur et al 2005). Such practices include short daily meetings as with Scrum (Highsmith and Cockburn 2001; Cockburn 2002A; Highsmith 2002) and XP (Cockburn 2002A), pair programming in XP (Highsmith 2002; Highsmith and Cockburn 2001), short-cycle user prototyping in DSDM, end of iteration process and team reviews in Crystal and ASD, and end of iteration process reviews with customer focus groups in Scrum and ASD (Highsmith and Cockburn 2001).

Agile methods also adopt certain technical practices that are largely aimed at improving quality and delivery. Code should be kept simple and technically advanced (Abrahamsson et al 2002). A simple design is advocated, i.e. the design should cover only what needs to be developed rather than cater for

future functions that might change anyway (Boehm and Turner 2005). Quality of design is stressed and design is done continually in smaller portions rather than all at once and upfront. Certain methods such as DSDM advocate the use of prototypes to address unstable or unknown areas such as new technology, business rules or user interfaces (Highsmith and Cockburn 2001). Ongoing design encompasses refactoring, where software is reorganised to remove duplication, simplify or add flexibility without changing functionality (Boehm and Turner 2005). Refactoring aids in constructing and maintaining a technically excellent product that can contribute significantly to creating business value (Highsmith 2002). XP advocates practices such as simple design, continuous integration, collective code ownership and refactoring with the aim of minimising time to create working code from elicited requirements (Turk et al 2005).

Some methods make use of pair programming. With pair programming two developers work together side by side on the same piece of work, a design, algorithm, code or a test (Boehm and Turner 2005).

Agile methods also make use of test-driven development where tests are written both before and during coding (Boehm and Turner 2005; Nerur et al 2005). There is also a greater focus on continuous testing (Highsmith 2002; Nerur et al 2005) as well as continuous integration of code (Nerur et al 2005).

c. Situations for using agile methods

Various authors (Beck 1999; Cockburn and Highsmith 2001; Wagner 2001; Boehm 2002; Highsmith 2002; Murthi 2002; Williams and Cockburn 2003; Little 2005; Nerur et al 2005) suggest that there are appropriate situations for the use of agile or traditional/plan-driven methods, labelled “home ground” by Boehm (2002). These situations are summarised in Table 1 below.

Table 1: Situations for agile and traditional methods

Factor	Traditional	Agile
Complexity		Simple (Little 2005)
Scope (requirements)	Well known early (Murthi 2002) Size well understood (Murthi 2002) Largely stable, will not change (Murthi 2002)	Vague/uncertain requirements (Murthi 2002) Unknown scope (Murthi 2002) Requirements subject to rapid change/volatile (Cockburn and Highsmith 2001; Murthi 2002; Highsmith 2002; Williams and Cockburn 2003) Market uncertainty (Little 2005)
Technology		Leading/bleeding-edge (Highsmith 2002; Murthi 2002)) Technical uncertainty (Little 2005) Fast-changing (Baskerville et al 2003)
Risks	Well understood (Murthi 2002) Minor impact (Murthi 2002)	Unknown risks (Murthi 2002) Major impact (Murthi 2002)
Primary objective	High assurance (Boehm 2002) Safety/mission critical (Williams and Cockburn 2003)	Rapid value (Boehm 2002) Not safety critical (Williams and Cockburn 2003)
Refactoring	Expensive (Boehm 2002)	Inexpensive (Boehm 2002)
Time	Clearly defined (Murthi 2002) Clear milestones (Murthi 2002) Long term (Baskerville et al 2003)	Not well defined/open (Murthi 2002) Unclear milestones (Murthi 2002) Subject to change (Murthi 2002) Short (Baskerville et al 2003)
Resources (money, infrastructure, people)	Approved and available (Murthi 2002) Has been done before (Murthi 2002) People familiar with tasks and tools (Murthi 2002)	Not fully approved or available (Murthi 2002) Need proof of concept (Murthi 2002) New skills needed (Murthi 2002)
Contracting arrangement	Fixed price, fixed budget upfront. Fixed time, price and scope. (Fowler 2005)	Fixed time and price, and allow scope to vary in a controlled manner (Fowler 2005)
Size	Larger teams (Boehm 2002) Larger products (Boehm 2002)	Smaller teams (Boehm 2002; Williams and Cockburn 2003) Smaller products (Boehm 2002)
Developers	Plan-oriented (Boehm 2002) Adequate skills (Boehm 2002) Access to external knowledge (Boehm 2002)	Agile (Boehm 2002; Fowler 2005) Knowledgeable (Boehm 2002; Williams and Cockburn 2003) Collocated (Boehm 2002; Williams and Cockburn 2003) Collaborative (Boehm 2002; Fowler 2005)
Customers	Access to knowledgeable, collaborative, representative, and empowered customers (Boehm 2002)	Collocated, dedicated, knowledgeable (technical and domain), collaborative, representative, and empowered customers (Boehm 2002))

Source: Adapted from Murthi (2002:16) and Boehm (2002:68)

Agile methods are seen to be more suitable for low complexity systems that are focused on delivering value rather than being mission-critical, where the requirements are uncertain and changing, where new technologies are being used and the cost of refactoring is relatively low. Agile projects are ideally those where timelines are short and changeable, the budget is variable and where scope might be the variable if time and price are fixed. Agile methods suit teams that are smaller with skilled developers and dedicated knowledgeable customers who are willing to work collaboratively in a collocated fashion.

A number of authors contend that there is not a 'one size fits all' approach and offer guidelines for determining the appropriate mix of traditional and agile approaches. De Marco and Boehm (2002) discuss the idea of a middle ground and the need for balance. Little (2005) describes a four quadrant grid based on complexity and uncertainty and suggests that high uncertainty low complexity projects would have greater success with agile methods.

Cockburn (2000) provides a grid for methodology selection, based on group size, system criticality and project priority. Moving across and up the grid to the right, methodologies incorporate more communication elements (catering for larger team size) and tighter controls (for more critical projects) resulting in greater cost.

Glass (2001) suggests that the best of agile and traditional methods can be combined. He considers project differences along the following dimensions: size (increasing needs greater formality), application domain (different business problems require different methods), criticality (increasing needs greater error removal), innovativeness of problem being solved (increasing needs more agile method), schedule (tightness may constrain choice of method) and people skills (lack thereof may constrain choice of method).

Boehm (2002) uses a risk driven approach, calculating risk exposure as a product of probability of loss and size of loss and plotting this as a function of investment in planning and processes, and as a function of market share erosion due to increased time to market. The intersection of these curves may fall into the sweet spot for agile, mainstream or plan-driven methods. By

examining a specific project in relation to the “home ground” situations, one can adapt the approach utilised and make use of a hybrid approach.

Theunissen (2003) advocates the use of a more agile approach for projects where requirements are susceptible to change, where the system can be delivered to users in an evolutionary way and where certain practices may be used to reduce project risks.

Mnkandla and Dwolatzky (2007) present a software-based selection tool that applies the following parameters in a selection matrix to confirm suitability for use of agile methods: Requirements Stability, Project Size, Development Timeframe, Project Complexity and Project Risk. Relevant practices can then be selected and tuned to the specific project environment.

The approaches discussed above suggest that the application of a software development method be it agile or not is driven by consideration of the specific situation at hand taking into account factors such as complexity, uncertainty, team size, criticality, priority, type of system, schedule and risk.

2.3.2 Comparing Agile and Traditional Methods

a. Differences

The use of words such as ‘manifesto’ and ‘extreme’ in the context of agile development might suggest that the agile development movement is rather radical, anti-establishment and completely at odds with traditional software development approaches. Whilst this section serves to highlight the differences between these approaches, the practical reality is that the different specific development methods provide a menu of choices which can be selected from and appropriately tailored and applied to specific situations.

A number of authors (Sutherland 2001; Boehm 2002; Highsmith 2002; Williams and Cockburn 2003; Lycett, Macredie, Patel and Paul 2003) contend that the nature of the software development process from the agile and traditional/engineering-based perspectives is fundamentally different. Agile proponents view software development as being an empirical or non-linear

process rather than the engineering-based view of software development as a defined, predictable and repeatable process. A defined process suggests that if predefined steps are followed the same results can be achieved each time. However, Williams and Cockburn (2003) assert that in the case of software development there are high levels of change in requirements, technology and people. Hence the process is not defined, as a set of predefined steps is unlikely to lead to desirable, predetermined outcomes. Traditional approaches assume that users are aware of what they need before seeing functioning software, and requirements can be specified in advance and will not change during development (Sutherland 2001).

A number of authors (Highsmith and Cockburn 2001; Cockburn 2002A; Lycett et al 2003; Fowler 2005; Nerur et al 2005) point out further specific differences in the emphasis of traditional versus agile approaches and these are summarised in Table 2.

Table 2: Traditional compared to agile software development

	Traditional	Agile
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning (Nerur et al 2005).	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change (Nerur et al 2005).
Project Cycle	Guided by/plan tasks or activities (Highsmith and Cockburn 2001; Nerur et al 2005) Sequential phases (Cusumano and Yoffie 1999) Project has defined start and end (Baskerville et al 2003)	Guided by/plan product features (Highsmith and Cockburn 2001; Nerur et al 2005) Parallel specification, development and testing (Cusumano and Yoffie 1999) Projects are an ongoing operation – small chunks can be rolled out (Baskerville et al 2003)
Development Model	Life cycle model (Waterfall, Spiral, or some variation) (Nerur et al 2005) Lifecycle specifies phases with defined outcomes (Nerur et al 2005)	The evolutionary-delivery model (Nerur et al 2005) Short iterative cycles of development (Nerur et al 2005)

	Traditional	Agile
Specification	Upfront specification and detailed design before coding (Cusumano and Yoffie 1999) All at once, upfront design (Highsmith and Cockburn 2001)	Specification evolves from initial vision (Cusumano and Yoffie 1999) Ongoing design in smaller chunks (Highsmith and Cockburn 2001)
Product release	Deliver product at end of development for customer evaluation (Ceschi et al 2005)	Frequent product releases, incrementally delivering functionality (Ceschi et al 2005)
Testing	Single integration and testing phase at end of project (Cusumano and Yoffie 1999)	Daily builds (Cusumano and Yoffie 1999)
Rules	Inclusive rules – externally defined predefined practices and conditions for every situation (Highsmith and Cockburn 2001)	Generative rules – minimum set of things for all situations that generate situation-appropriate practices. Depend on individuals and creativity not written rules to solve problems (Highsmith and Cockburn 2001).
Control	Process centric (Nerur et al 2005; Fowler 2005) Focus is on highly codified, optimised and repeatable processes (Nerur et al 2005). Explicitly defined, standardised processes (Lycett et al 2003). Move software process control from producers to managers (Lycett et al 2003). Standardise people to the process (Cockburn and Highsmith 2001; Highsmith 2002)	People centric (Abrahamsson 2002; Nerur et al 2005; Fowler 2005). People are a key driver for project success (Highsmith and Cockburn 2001; Fowler 2005) Process control with producers; management has coordination role (Lycett et al 2003). Capitalise on individual and team strengths and adapt process (Cockburn and Highsmith 2001; Highsmith 2002)
Role Assignment	Individual – favours specialization (Nerur et al 2005)	Self-organising teams – encourages role interchangeability (Nerur et al 2005)
Knowledge Management	Explicit, documented (Boehm 2002; Nerur et al 2005) External (Cockburn 2002A; Lycett et al 2003)	Tacit (Boehm 2002; Cockburn 2002A, Nerur et al 2005) Embodied in the team (Boehm 2002), peer-based collective learning (Lycett et al 2003)

	Traditional	Agile
Communication	Formal (Nerur et al 2005) Written (Highsmith and Cockburn 2001; Cockburn 2002A; Highsmith 2002)	Informal (Cockburn 2002A; Baskerville et al 2003; Nerur et al 2005) Verbal, face to face (Highsmith and Cockburn 2001; Highsmith 2002; Lycett et al 2003)
Management role	Planning, executing and controlling (Lycett et al 2003; Nerur et al 2005)	Design, coordination, enabling of activities. Shaping environment for action (Lycett et al 2003). Facilitation, coordination of collaborative efforts and group decision-making (Nerur et al 2005)
Management Style	Command-and-control (Cockburn and Highsmith 2001; Nerur et al 2005)	Leadership-and-collaboration (Cockburn and Highsmith 2001; Nerur et al 2005)
Customer's Role	Important (Nerur et al 2005) Customer input used as input for future projects (Cusumano and Yoffie 1999)	Critical (Nerur et al 2005) Customer feedback considered during development (Cusumano and Yoffie 1999)
Measure of project success	On-time and on-cost (Fowler 2005)	Customer value of software greater than cost (Fowler 2005).

Source: Adapted from Nerur et al (2005:75)

Traditional approaches are also seen to be plan-driven with a focus on producing documented architectures and plans. However that does not mean that agile approaches are akin to cowboy style development or hacking. Planning is important in agile methods, but more value is placed on the planning process rather than producing planning documentation (Boehm 2002). In a method such as XP, continuous and complicated planning takes place on a daily basis (DeMarco and Boehm 2002).

Agile methods adopt a cyclical development model with parallel rather than sequential phases characterised by traditional waterfall type approaches. Upfront specification in agile methods is not expected and the design typically evolves through the course of the project. There are regular product releases each with their own testing phase rather than a single testing and delivery phase at the end of the project.

Agile methods tend to be people rather than process-focused, relying on people and their creativity. Team members organise themselves and may rotate roles rather than operate only within a defined specialisation. While traditional methods emphasise documented knowledge and formal written communication, agile methods tend to rely more on tacit knowledge and informal face to face communication between team members.

The differences in the way agile teams operate require a different management role and style. The agile manager plays a more facilitative and enabling role rather than having the traditional emphasis on planning, execution and control.

A more collaborative management style is required for agile projects. Cockburn (2002A:12) highlights the differences in the traits that agile project managers rely on that allow the team to move and change direction rapidly: “An agile project manager relies on discipline, skill, and understanding, while requiring less formality, process, and documentation.”

While traditional methods acknowledge the role of the customer as being important, agile methods treat the customer as central with customer feedback considered throughout the project.

Traditional methods rate the success of projects in terms of delivery within time and cost where agile methods are more concerned about customer value. According to Baskerville et al (2003), agile methods tend to place a greater emphasis on speed, responsiveness and creativity as opposed to traditional priorities of quality or cost.

b. Similarities

While the above discussion highlights the differences between agile and traditional approaches, the overall goals are common: satisfying customers (Wagner 2001) including their need for quality and flexibility (Lycett et al 2003), producing the intended software product (Glass 2001; Lycett et al 2003) and meeting internal needs such as cost (Lycett et al 2003). Traditionalists and agilists want to perform software engineering in a better manner and produce better products (Glass 2001).

A number of authors (Beck 1999; Abrahamsson et al 2002; Williams and Cockburn 2003; Theunissen 2003) point out that the ideas and practices that agile methods bring about are not new. They claim that agile methods provide a different way of approaching software engineering together with the theoretical and practical framework. The realisation of the importance of focusing on people in order to improve software quality and productivity is also not new (Beck 1999; Abrahamsson et al 2002; Williams 2004). Abrahamsson et al (2002) observe that agile software development approaches embody much of what current software development practice entails whilst Theunissen (2003) suggests that some of the agile practices could be described as common sense. Hilkka, Tuure and Matti (2005) conclude that XP is a new name for an old way of working for in-house software development. It formalises habits that occur naturally in specific environments – close customer involvement, quick releases, cyclical development and rapid response to changes.

Planning must take place on any project whilst at the same time recognising that in reality requirements will change; agile methods accommodate change, other methods still plan and manage it (Glass 2001; Baskerville et al 2003).

Both agile and traditional methods emphasise the need to develop towards a "just-right" requirements set (Baskerville et al 2003).

Team reflection and changes in behaviour to improve effectiveness is characteristic of agile methods but process change and improvement are also incorporated in the CMM's (Capability Maturity Model) highest maturity level (Glass 2001; Boehm and Turner 2005).

Documentation is important in all methods, i.e. user manuals, requirements specifications and maintenance documentation to support the ongoing product development (Glass 2001). Agile methods provide for the use of documentation but are against the overemphasis on documentation that does not support the primary goal of delivering effective software (Theunissen et al 2003).

Collaboration is needed but is not an alternative to a contract (Glass 2001; Baskerville et al 2003).

Agile methods emphasise engaging the customer but traditional methods do also focus on the need to understand the customer problem (Baskerville et al 2003).

2.4 Attitude towards agile software development

Attitudes toward agile development may be reflected in views of the benefits of using agile methods compared to shortcomings of agile methods and the challenges associated with implementing agile software development (depending on how these might be overcome).

2.4.1 Benefits of agile software development

Various authors cite benefits and reasons for organisations adopting agile methods of development.

A number of authors contend that agile software development contributes towards improved software product quality (Cockburn and Highsmith 2001; Lindvall et al 2004; Ceschi et al 2005; Melnik and Maurer 2005; Schatz and Abdelshafi 2005; Ambler 2006; Begel and Nagappan 2007; Ambler 2008; Chow and Cao 2008; Dyba and Dingsøyr 2008) in terms of reduced software defects (Schatz and Abdelshafi 2005; Begel and Nagappan 2007; VersionOne Inc. 2007; VersionOne Inc. 2008), a more stable set of features (Begel and Nagappan 2007) and improved test coverage (Lindvall et al 2004). Chow and Cao (2008) contend that agile oriented project management, requirements management and configuration management processes together with daily face to face meetings, good progress tracking and regular working schedules can contribute towards improved quality.

The improved quality of software can also be attributed to practices such as pair programming (tends to promote simpler design and prevent “gold plating”) (Lindvall et al 2004), automated testing (Lindvall et al 2004; Melnik and Maurer 2005; Begel and Nagappan 2007), test-driven development (Begel and Nagappan 2007) and ongoing refactoring (Begel and Nagappan 2007; Chow and Cao 2008). Lindvall et al (2004) also mention improved quality of changes

that are implemented due to continuous integration. Chow and Cao (2008) emphasise the importance of the team environment in terms of promoting quality particularly through collocation, self-organisation and small teams.

Various authors (Cockburn and Highsmith 2001; Ceschi et al 2005; Ambler 2006; Ferreira and Cohen 2008; Ambler 2008) believe that agile software development facilitates better customer satisfaction. Increased satisfaction can come from customers' needs being met (Murthi 2002). Chow and Cao (2008) refer to agile software engineering techniques that contribute positively to delivering a good working product (quality) and meeting customer requirements (scope) such as well-defined coding standards, simple design, the right amount of documentation and integration testing. Technical practices such as collective code ownership, continuous integration and test-driven development also contribute towards the development of better systems and thus better customer satisfaction (Ferreira and Cohen 2008). Other factors contributing towards achieving a solution that meets customer requirements include customer involvement where there is a good customer relationship, a strong level of commitment, high presence and full decision-making authority. The agile delivery strategy is also important for customer satisfaction i.e. regular software delivery and delivering important features first (Chow and Cao 2008).

Increased satisfaction also comes from customers receiving rapid value (Boehm and Turner 2005; Ceschi et al 2005; Dyba and Dingsøyr 2008) with early production and more responsive software development (Fowler 2005; Ferreira and Cohen 2008) due to the iterative approach (Ferreira and Cohen 2008). Customers are able to provide feedback and see changes being incorporated more easily (Dyba and Dingsøyr 2008; Ferreira and Cohen 2008).

The level of customer and project team communication is improved through increased customer presence and use of the "planning game" (Lindvall et al 2004). There is also improved alignment between Information Technology (IT) and business goals (VersionOne Inc. 2007), better customer focus (Begel and Nagappan 2007), better collaboration (Dyba and Dingsøyr 2008) and improved prioritisation (Baskerville et al 2003; Begel and Nagappan 2007). Customers are also able to obtain prompt feedback on the impact of changes in

requirements on costs, schedule (Baskerville et al 2003) and remaining workload (Schatz and Abdelshafi 2005). The close interaction between the team and the product owner allows the customer to see the product evolve and promotes increased confidence in the team's ability to deliver value with the specified timeframes (Schatz and Abdelshafi 2005). Ferreira and Cohen (2008) find that the level of stakeholder satisfaction with the development processes influences their satisfaction with regard to the project outcome, the system quality and the extent to which they feel the system meets their needs.

Software is provided in quick releases (Begel and Nagappan 2007). This shortened release cycle (weeks instead of months or years) facilitates progress tracking, quality monitoring, feature evaluation and product improvement. Bug-fixing turnaround times are also improved (Begel and Nagappan 2007). As slices of functionality are implemented per release, product owners are able to review features and focus on highest value items; remaining items can be dropped if deemed unnecessary or have their priority lowered (Schatz and Abdelshafi 2005). The agile delivery strategy can have a positive impact on project timeliness (Chow and Cao 2008). In addition the shorter cycle provides an accelerated time to market (Schatz and Abdelshafi 2005; VersionOne Inc. 2007; VersionOne Inc. 2008).

The agility in agile software development methods is reflected in faster responses to change requests (Lindvall et al 2004; Begel and Nagappan 2007) and an enhanced ability to manage changing priorities (VersionOne Inc. 2008). This ability to deal with change comes from flexibility in both the design and in the development process itself (Begel and Nagappan 2007).

Increased productivity (Lindvall et al 2004; Melnik and Maurer 2005; Ambler 2006; Begel and Nagappan 2007; VersionOne Inc. 2007; Ambler 2008; VersionOne Inc. 2008) is also seen as a benefit of agile software development and can be demonstrated by increased lines of code per hour (Dyba and Dingsøyr 2008).

Agile software development can make management easier in a number of ways. Measurement of progress is improved by using working software as the

indicator (Glass 2001; Boehm and Turner 2005); hence there is greater visibility of the project's true state (Fowler 2005; VersionOne Inc. 2007; VersionOne Inc. 2008). Requirements are better managed (Ceschi et al 2005), especially where these are emerging, and misconceptions can also be identified earlier in the process (Boehm and Turner 2005). Work processes for managing defects are also improved (Dyba and Dingsøyr 2008).

Agile methods are seen to have reduced process complexity (VersionOne Inc. 2007; VersionOne Inc. 2008). Begel and Nagappan (2007) label agile as a "more reasonable process" where less time is wasted on tasks deemed irrelevant by developers and the level of documentation and planning is "just-in-time" and sufficient for the following iteration. They find agile methods more manageable, less bureaucratic, more dynamic and having less overhead.

Several authors discuss the positive aspects of agile software development in relation to the software development team and the individuals involved. Begel and Nagappan (2007) note the improved communication and coordination among team members. The daily scrums (meetings) bring developers and testers together, enhance peoples' awareness of other team members' activities and promote earlier identification and resolution of development issues. These stand-up meetings also contribute towards improved work discipline (Lindvall et al 2004). Team collocation can also be a positive factor in promoting good teamwork (Schatz and Abdelshafi 2005).

Projects may be less vulnerable to individual team members where pair programming is used (Murthi 2002). Information and knowledge transfer within the team may also be improved through pair programming (Lindvall et al 2004; Dyba and Dingsøyr 2008). The use of agile methods may also contribute towards a shortened learning curve for new developers (Lindvall et al 2004).

Agile methods support greater empowerment of developers/individuals through realistic goals, more rapid feedback cycles, ownership and flexibility (Boehm and Turner 2005). Schatz and Abdelshafi (2005) in their experience of implementing Scrum report that Scrum allows the team to establish a long term sustainable pace, an improved working environment with lower staff turnover,

increased level of trust and an increased sense of developer ownership and pride in their work. The close working relationship between the development team and product owner allows developers to obtain a better understanding of the business value of features being implemented and increases their level of influence on the product being developed (Schatz and Abdelshafi 2005). Developer skills in communication, commitment, cooperation and adaptability are improved (Melnik and Maurer 2005). Employee morale is higher (Cockburn and Highsmith 2001; Lindvall et al 2004; Begel and Nagappan 2007; VersionOne Inc. 2007) and team satisfaction is improved (Ceschi et al 2005; Dyba and Dingsøyr 2008).

Finally it is suggested that costs may be reduced (VersionOne Inc. 2007; Ambler 2008; VersionOne Inc. 2008). Cost savings could be attributed to less detailed specifications with decreased specification updates (Lindvall et al 2004), a high level of team capability and also the agile delivery strategy (Chow and Cao 2008). However, in some cases there could also be no benefit or a negative cost impact (Ambler 2008; VersionOne Inc. 2008).

2.4.2 Shortcomings of agile software development

The nature of agile software development methods and some of the specific characteristics and practices can create some inherent limitations of such methods.

Agile methods are seen to offer limited support in environments where team members and customers are physically distributed (Ceschi et al 2005; Begel and Nagappan 2007), as this geographical separation does not directly cater for the level of face to face communication required for agile software development (Turk et al 2002). According to Grossman et al (2004) effective ways of using agile practices such as pair programming, sharing user stories, daily stand-up meetings, release planning meetings as well as dealing with off-site customers in distributed environments are not extensively understood or used. Face to face communication can be accommodated to some extent with the use of technology such as video-conferencing or through planned periodic face to face meetings (Turk et al 2002). Documentation of requirements and designs would

be needed to ensure that there is a consistent vision of the project across the team (Turk et al 2002; 2005).

Agile methods also tend to have limited support for large teams (Cockburn and Highsmith 2001; Ceschi et al 2005; Begel and Nagappan 2007), as large teams require more interactions and a greater management focus; also the size of teams can inhibit effectiveness and frequency of face to face communication. More documentation may be needed to coordinate between many team members (Turk et al 2005).

Dyba and Dingsøyr (2008) suggest that agile methods may not be well suited to large and complex projects. Particularly in complex systems where there are numerous system interfaces required even for basic services, the high level of dependencies may make it impossible to break work into small work packages that can be done in iterations (Turk et al 2005).

Legacy systems may also not be conducive to refactoring or disassembly for incremental development (Boehm and Turner 2005). Nerur et al (2005) contend that organisations using mainframe technologies may find it more difficult to incorporate agile methods than those using object-oriented development.

In situations where there are multiple teams agile projects may be difficult to coordinate, particularly where other teams may not understand agile scheduling practices. There may also be increased difficulty, cost and time for integrating modules developed by different teams (Begel and Nagappan 2007).

Within agile teams there is a high level of dependence on individual developers in terms of domain knowledge, proficiency in technology and tools being used and communication skills (Baskerville et al 2003; Hilkka et al 2005). This means that the team members are less interchangeable which can impact how projects are managed (Dyba and Dingsøyr 2008).

Development outsourcing or sub-contracting is also viewed as not conducive to applying agile methods, as typical sub-contracting arrangements require sub-contractor deliverables to be precisely stipulated. More flexible contracts would

be required (Turk et al 2002; 2005). Flexible contracts are used because rapid responses to change make it difficult to maintain contract variables (scope, price, time) fixed over the duration of a project (Ceschi et al 2005).

In general the lack of a fixed and predictable schedule for agile projects may be problematic (Begel and Nagappan 2007).

Some of the agile practices themselves are seen to have inherent limitations. There can be issues with design, such as lack of up-front design and poor design (Begel and Nagappan 2007; Dyba and Dingsøyr 2008). Since agile processes focus on building products for a specific problem rather than a generalised solution or reusable software there is limited support for reusability (Turk et al 2002; 2005).

Agile methods (particularly XP and to a lesser extent Crystal and DSDM) tend to have a lesser focus on architectural preplanning, the approach being to cater for architectural features that support the current version rather than for future requirements. This approach is limiting in situations where future requirements are known and architectural support is not provided for these early on (Boehm 2002). A good design should anticipate future customer enhancements and a more complicated upfront design may be able to simplify ongoing maintenance and better accommodate future changes (Glass 2002). For large systems a lack of attention to architectural elements for performance, fault tolerance and security can create the need for extensive rework in future (Boehm 2002).

Refactoring can consume increasing effort as the number of requirements increases, particularly with less competent developers (Boehm 2002).

There is a lesser focus on maintenance and a lack of documentation on agile projects can make maintenance time consuming (Baskerville et al 2003). Where there are complex systems with long lifecycles and such systems need to be modified, the lack of models describing these systems may force developers to analyse the code in order to understand these systems (Turk et al 2005). The relative lack of documentation may also result in the loss of key information related to requirements, even where the customer is present on site (Haynes and Friedenberg 2006).

The reliance on tacit knowledge may also mean that gaps in knowledge are not picked up and a possible consequence is that the team may make irrecoverable architectural errors (Boehm 2002). Furthermore the focus on tacit knowledge may lead to corporate memory loss and impede the organisation from learning from past knowledge (Turk et al 2005).

Agile process quality control methods are not seen to be sufficient to support safety-critical software and more formal quality control techniques might be required (Turk et al 2002; 2005). Williams (2004) also notes that quantitative quality metrics as well as process modelling and measurements are not prominent in agile methods.

A potential problem in agile methods can be the ability to accommodate corrections within the same lifecycle where deficiencies are identified. Organisations that do not have mature software engineering environments may struggle to implement the necessary feedback loops and react accordingly (Williams 2004).

Begel and Nagappan (2007) suggest that agile methods require too many meetings and are specifically concerned around daily meetings (Scrum) and inefficiency of meetings if run poorly. Furthermore they suggest that the immediate focus on deliverables for the day could cause teams to lose sight of the bigger picture.

2.4.3 Challenges

The implementation of agile methods into a software development organisation is a change process that affects multiple stakeholders (customers, development staff and managers), the manner in which they engage with each other, and they way in which they execute their work on a day to day basis on projects. These changes present a number of challenges when implementing agile methods and there are various factors that can assist in transitioning to the use of agile methods successfully.

Most software projects in large organisations are not undertaken with small collocated teams and agile methods need to address scalability concerns if they are to be seriously considered (Abrahamsson et al 2002).

Ramesh, Cao, Mohan and Xu (2006) discuss a number of challenges that can be experienced in agile distributed development. Based on their research they offer some practices intended to address these challenges.

Communication in a distributed team can present specific challenges: starting communication can be more difficult; there is a potential for misunderstanding; frequency of communication can be lower; cost of communication can be higher and there can be time differences. Agile development relies more on informal interactions rather than documentation where distributed teams require more formal mechanisms including detailed architectural design and planning. Useful practices to overcome this challenge can be: synchronising team working hours; using informal communication but still through formal channels such as the project lead; having constant communication such as daily meetings, online chatting, SMS (Short Message Service), round-the-clock use of Blackberries by project leads; and weekly video-conferencing with senior managers. Knowledge sharing can be facilitated with processes and tools including a product and process repository (Ramesh et al 2006).

Controlling the process and quality can be more complicated in distributed teams. Agile methods are more people-oriented with informal processes whereas distributed environments have more formal processes for greater control. Agile development tends to have an ongoing feedback and negotiation of quality with the customer while distributed environments will often have fixed upfront commitments regarding quality. Practices adopted to address this challenge include using onshore quality assurance and participation in design reviews and using documentation (sometimes developed after actual work is completed) to supplement informal communication. More upfront attention should be given to finalising critical requirements and developing a high level architecture with the use of documented short use cases and user stories (Ramesh et al 2006).

With distributed teams there can be a lower level of trust between the members of the distributed team and potentially lower team morale. Agile environments typically have loosely defined contracts where distributed environments will have more explicit targets and milestones and more detailed requirements. Agile environments require cohesive teams with a high level of trust. Frequent visits by distributed partners as well as visits by sponsors for finalising the contracts and to evaluate projects can help to build a greater level of trust. Teams can be selected with members who have prior working relationships and who together as a team have the required expertise (Ramesh et al 2006).

Challenges that might be experienced in implementing agile methods and reasons for lack of adoption of agile methods include resistance from the various parties involved. Large, mature organisations may resist extensive changes in working practice and this can pose a considerable barrier to implementation (Lycett et al 2003).

Developers may resist agile methods (Ceschi et al 2005) for a number of reasons. Some may be used to working alone (Nerur et al 2005) whilst some enjoy producing non-code artefacts (Cohn and Ford 2003). Others may perceive agile methods as being less productive, hurting their development style, stifling creativity and individual growth (Beck 1999), or a form of micro-management (Cohn and Ford 2003; Williams and Cockburn 2003). Agile development requires more interpersonal skills, patience and adoption from the entire team (Begel and Nagappan 2007) and there may be some programmers who don't want to embrace the social aspects and don't want to leave old habits (Beck 1999).

A number of authors (Cockburn and Highsmith 2001; Cohn and Ford 2003; Boehm and Turner 2005; Fowler 2005; Nerur et al 2005; Chow and Cao 2008; Dyba and Dingsøyr 2008) stress the importance of having developers with high levels of individual competence as an important contributor to success. Agile methods place high value on the individuals involved in projects who need to have the appropriate technical and interpersonal skills to accept high levels of responsibility and decision-making working in self-organising teams.

The development team may require training in new skill sets (Murthi 2002; Nerur et al 2005; Chow and Cao 2008), both technical and non-technical, e.g. use of tools, refactoring and configuration management (Nerur et al 2005), meeting management skills to run short, tightly focussed meetings (Rising and Janoff 2000), "soft skills" such as cost estimation, time budgeting, risk identification, contingency planning (Murthi 2002), customer involvement (Van Deursen 2001), and negotiation (Schatz and Abdelshafi 2005).

The role of testers is also somewhat changed in agile projects, and testers may view the increased attention on testing as micro-management since testing occurs at each iteration and testers work closely with developers earlier in the project (Cohn and Ford 2003).

In addition to resistance from team members agile methods may also face resistance from managers who consider agile development ill-disciplined, i.e. an opportunity for developers to hack (Williams and Cockburn 2003). They may view the lack of detailed preliminary cost evaluation in projects as a problem (Boehm and Turner 2005; Ceschi et al 2005). Progress measurement on agile projects differs from the traditional approach (Boehm and Turner 2005) but managers might still want reporting or progress measurements that are not well supported in the agile environment (Cohn and Ford 2003; Begel and Nagappan 2007).

Contracts, milestones and progress management techniques must be adapted to suit the pace of agile methods (Boehm and Turner 2005). Ceschi et al (2005) suggest the use of more flexible contracts versus fixed contracts (with predefined functionality, price and time). Such a change requires commitment and buy-in from both upper management and the customer towards the agile development approach. There is a shift from the traditional control mechanisms towards producing working code. Middle and upper management require a cultural shift and must change the way the organisation functions and put more trust in the development team (Abrahamsson et al 2002).

Agile methods may also clash with accepted traditional approaches as there is a lower or more informal emphasis on non-functional requirements, documentation, milestones and formal requirements (Boehm and Turner 2005).

Management may also struggle with the timing on agile projects; they may have difficulty with lack of predictability regarding when specific features would be produced (Begel and Nagappan 2007) or when the project will end (Cohn and Ford 2003; Schatz and Abdelshafi 2005). As requirements change from iteration to iteration, determining how far one is from the product release can be difficult (Schatz and Abdelshafi 2005).

Human Resources (HR) departments and managers may find agile methods different to what they are used to. Agile team members cross traditional job descriptions and may require additional skills and experience to perform (Boehm and Turner 2005). People empowerment and devolved decision-making promoted by agile methods may conflict with existing HR-based procedures (Boehm and Turner 2005), which may have to be adapted accordingly. Practices such as pair programming might also initially be considered strange or unproductive by HR (Cohn and Ford 2003). Cohn and Ford (2003) and Boehm and Turner (2005) propose involving the Human Resources department and addressing their issues at the very outset, as this also ensures they are in a position to address complaints or questions from staff regarding new practices (Cohn and Ford 2003).

Organisations need a culture change in order to convince management of the benefits of agile development (Lycett et al 2003). If the paradigm shift is too great and supporting adoption models and techniques are absent then organisations and developers are unlikely to adopt agile methods (Abrahamsson et al 2002). A culture that is at odds with agile values may cause an agile project to fail (VersionOne Inc. 2008). The organisational culture should support the human-centred view espoused by agile development. Organisational culture traits emphasised in the literature are trust, respect (Nerur et al 2005), people-centred, and collaborative (Cockburn and Highsmith 2001). The culture should be supported by an organisational form that emphasises autonomy and co-operation (Nerur et al 2005).

Boehm and Turner (2005) mention that there are still some myths or misperceptions about agile methods, including views that agile methods are not adequate for managing defects, are a fad, or that agile projects are not managed.

Customers need to participate actively throughout the development process and should be in close proximity (Nerur et al 2005) for higher levels of accessibility and collaboration. There may be resistance from customers (Ceschi et al 2005) particularly towards high levels of interaction, feedback and input (Boehm and Turner 2005). Uncooperative customers may not want to fully participate in defining tests, priorities and requirements (Beck 1999).

Some suggestions to improve the effectiveness of customer involvement in agile development projects include: setting up a customer team representing various stakeholders instead of a single person (Van Deursen 2001), ensuring representation from both executive/management level (to ensure strategic fit and project resource commitment) and a target user group ideally with some IT knowledge to act as subject matter experts (Haynes and Friedenberg 2006). Customer education is required (Boehm and Turner 2005), specifically training on what is expected from customers and skills/techniques to be used e.g. writing requirements, facilitation and coordination (Van Deursen 2001). The customer must have authority to make decisions in the event of stakeholder conflicts and must possess adequate knowledge of the domain (Van Deursen 2001).

A lack of user and executive support can be detrimental to a project (Cockburn and Highsmith 2001). Some authors (Cockburn and Highsmith 2001; Grossman et al 2004; Schatz and Abdelshafi 2005) believe that executive support is essential to sustain the change in the implementation of agile methods whilst Chow and Cao (2008) found executive support not to be critical for success on agile projects.

Agile methods may be less conducive to implementation of process standards (e.g. CMMi or ISO), ratings or certification (Lycett et al 2003; Boehm and Turner 2005), as agile methods do not cater for the extent of documentation and

infrastructure required for some of the levels of certification (Boehm and Turner 2005). However, agile methods can be adapted to comply with standards such as ISO 12207:1995 for software lifecycle processes (Theunissen et al 2003).

According to Grossman et al (2004), it is not apparent how to integrate a methodology such as XP into governance models within large organisations. Such models usually try to exercise control over budgets and alignment with IT strategies. At each phase projects must go through an executive decision checkpoint for approval to proceed to the next phase. Agile methods' practice of continuous refactoring may conflict with existing processes and structures such as change control boards (Lindvall et al 2004).

In larger organisations where cross-team communication is required there can be increased communication overhead and cultural conflict. There can also be double work where new practices are not well integrated with traditional processes and where agile teams need to comply with traditional processes, particularly in the areas of planning, requirements management, acceptance test management, quality management and documentation (Lindvall et al 2004).

There can also be risk associated with changing the development approach itself. However, this risk is minimised as schedule slippage could be detected early since the agile development cycles are short (Abrahamsson et al 2002).

Dyba and Dingsøyr (2008) note that using some of the agile practices themselves may present challenges: The on-site customer role may be stressful and unsustainable over extended periods. Where customer representatives are available but do not have a good sense of direction, agile developers may follow but the end result might be an inferior system (Highsmith and Cockburn 2001). There may also be misunderstandings between the customer and developers due to their differing use of language and developers may then make decisions on behalf of the client resulting in sub-optimal solutions (Gerber et al 2007). Where the customer is absent system analysts may need to represent the customer's interests and requirements and where they lack authority to enforce decisions, developers may drive decisions based on technology rather than on business need (Gerber et al 2007). Pair

programming may be strenuous because of the level of interaction (Theunissen 2003) and intense concentration (Dyba and Dingsøyr 2008). It may also be difficult where there is a great skill differential between the members of pairs. Test-first development may also be difficult (Dyba and Dingsøyr 2008).

Agile practices require appropriate technology and tools to support their use (Nerur et al 2005), e.g. the infrastructure to support continuous software integration (Grossman et al 2004; Madsen 2005; Haynes and Friedenberg 2006), automated testing (Grossman et al 2004), iterative development, configuration management and refactoring (Nerur et al 2005).

Murthi (2002) cautions organisations to be aware of people and team stress levels as they impact communication and cooperation negatively. Tempo, "the pace at which development activities proceed" (Haynes and Friedenberg 2006:18), should not necessarily be fast-paced, but appropriate to the project complexity, team expertise and comfort level, and the rate at which requirements are emerging. A number of authors speak to the importance of trying to maintain a sustainable pace without excessive overtime (Grossman et al 2004; Madsen 2005; Schatz and Abdelshafi 2005).

Agile projects may also focus too much on short-term deliverables. Sometimes teams can lose sight of the base technical infrastructure and longer term maintainability (Schatz and Abdelshafi 2005). Agile methods' focus on simple design and short iterations may lead to architectural deficiencies in the developed software. Lack of documentation of the architecture and design can negatively affect future maintainability; a situation can arise where discrepancies between system requirements are not picked up early (Gerber et al 2007). To combat this, a stable extensible architecture is important (Schatz and Abdelshafi 2005) and an expert team member should be given the task of ensuring coherence of overall system architecture (Haynes and Friedenberg 2006). A simple design, although guided by the functionality required immediately, should also incorporate what is required to build a stable, scalable product (Grossman et al 2004).

There may also be challenges in dealing with stakeholder feedback during reviews. Stakeholders might make suggestions that developers act on as new requirements while stakeholders may not be expecting these to be taken literally and may not be aware that they are being actioned. The addition of items to the product backlog needs to be carefully controlled with product owners stipulating the priority of additional requirements (Schatz and Abdelshafi 2005).

Other challenges noted are culture clashes during adoption (Begel and Nagappan 2007) and superficial knowledge of the topic of agile methods (Ceschi et al 2005). Lack of knowledge of agile methods can cause projects to fail (VersionOne Inc. 2008). It is important for managers to be knowledgeable about agile methods (Boehm and Turner 2005). Some companies introducing agile methods have made use of external consultants with expertise in the agile development practices and experience in using them. The use of external consultants can provide an independent opinion and new ideas (Murthi 2002). Mentors who have themselves made mistakes can advise on how mistakes can be avoided (Fowler 2005). External coaches can provide honest objective feedback and coaching can serve as a means to enforce a culture of learning (Schatz and Abdelshafi 2005).

A critical factor in adopting an agile method is “being agile, or adaptive in the adoption process itself” (Haynes and Friedenberg 2006:15). The methodology should evolve to fit the organisational culture (Haynes and Friedenberg 2006). Various authors emphasise the selection and use of an agile method that is appropriate to the organisation i.e. that fits in with the organisation’s priorities (Boehm and Turner 2005), operating objectives, development team members and the specifics of the development project (Haynes and Friedenberg 2006). The method adopted should take cognisance of current practices (Boehm and Turner 2005; Nerur et al 2005) especially if these have proven useful (Haynes and Friedenberg 2006). It is important that the methods adopted are tailored to meet the unique requirements of the team (Cockburn and Highsmith 2001; Murthi 2002). Boehm and Turner (2005) suggest using only the process

aspects of the methods that seem indispensable and building up a process from the bottom up.

Lindvall et al (2004) advocate tailoring the agile method (XP in the cases examined), recognising the software development projects are interdependent with and must follow the rules of the overall organisation. New practices must co-exist with existing practices that they might conflict or overlap with. In larger organisations the project team must interact with other teams and fit in with the organisation's standard processes and quality systems.

Dyba and Dingsøyr (2008) suggest that traditional project management models should be integrated with agile project management. Projects' characteristics should be compared with specific agile methods' requisite characteristics.

Boehm and Turner (2005) and Haynes and Friedenberg (2006) advocate a risk-based approach. The risks should be evaluated in order to determine the level of agility that is sufficient, taking into consideration the risks of too much/little agility (Boehm and Turner 2005). The various aspects of the method can be considered in relation to the risk profile of the operating environment, and they may be tailored, relaxed, or enforced depending on the project/component-specific risk level (Haynes and Friedenberg 2006).

In considering the factors above, Cohn and Ford (2003) contend that approaches to adopting agile methods that work in one case may not work in another. They expect the understanding of best practices to rise with increased uptake. Rising and Janoff (2000) note that performance gains come with experience and that development teams need to be given some time to first gain experience. In the early stages, Boehm and Turner (2005) suggest the results of pilot projects must be rewarded, and appreciation shown regardless of the outcome. Finally the adoption of agile development is not an outcome but a continual learning process: Metrics should be captured and lessons learnt shared (Boehm and Turner 2005) and organisations should learn from their experiences and strive to continually improve (Schatz and Abdelshafi 2005).

2.5 Conclusion of Literature Review

Agile software development methods can be defined as a set of methods that are adaptive, flexible, lightweight and people-centred, delivering solutions with self-organising, collaborative teams using an iterative, incremental approach to continually respond to changing environments and stakeholder needs. Agile methods have their origins in various approaches that have been in use since the 1980's and have matured over two decades, culminating in the gathering of software practitioners in 2001 which sought to derive a common foundation for this group of methods. The main agile methods are: XP, Scrum, ASD, DSDM, the Crystal Family of Methodologies, and FDD with commonly used methods being XP, Scrum and MSF for Agile.

The four values and twelve principles of agile methods set out in the 2001 Agile Manifesto form the framework for describing agile methods. Agile methods are characterised by short, iterative, feature-driven, time-boxed development cycles with customers being actively involved in providing continual feedback and dynamically prioritising their requirements. Teams are typically collocated and small with team roles being flexible and teams organising themselves. Documentation is minimised with a high reliance on tacit knowledge. Some of the agile practices include: daily meetings, simple designs, prototyping, refactoring, continuous integration, pair programming and test-driven development. Agile methods are seen to be suitable for specific types of systems, projects and environments.

The nature of agile methods is different to traditional software development methods largely arising from the fundamental view of software development as an empirical or non-linear process. However, generically agile and other methods share the same broad objectives.

Some of the benefits of using agile methods include: improved quality; better customer satisfaction; better alignment between IT and business goals; shortened release cycles with accelerated time to market; faster responses to change requests; increased productivity; easier management; greater visibility;

reduced process complexity; improved communication and coordination; greater empowerment of individuals and potentially reduced costs.

However, there are a number of shortcomings associated with agile methods: lack of a fixed and predictable schedule; lesser focus on architectural planning; lesser focus on maintenance; lack of documentation and reliance on tacit knowledge. Also agile methods offer limited support for: physically distributed teams; large teams; large complex projects and multi-team projects.

Implementing agile methods in practice can present some challenges since most software projects in large organisations are not undertaken with small collocated teams. Some of the challenges associated with agile distributed development are communication, process and quality control and trust issues. Initial implementation can face resistance from various parties such as developers, testers, managers and customers. Agile methods can conflict with accepted traditional approaches and governance models and culture clashes may arise during adoption. Knowledge of agile methods may also be limited. Whilst these might be seen as challenges there are potentially various ways that some of these challenges can be overcome and factors that can be taken into account to promote more successful implementation of agile methods.

As mentioned in Section 2.2.1, a single universal definition of agile software development is not presented in the literature. Agile software development may often be described with reference to its underlying values and principles. It is also often described by comparing and contrasting agile methods with traditional/plan-based/engineering methods of software development (see Section 2.3.2), or with hacking/cowboy-style development.

Therefore a complete and common understanding of this phenomenon cannot be assumed. Hence it is important to assess first what is understood by agile software development before exploring other dimensions. Furthermore, the participants' understanding may also influence their attitude towards agile methods.

The attitude towards agile software development may not be pre-judged. It may be favourable, unfavourable, indifferent, sceptical, ambivalent, or something

else. These attitudes may originate from participants' reading, what they have heard from others, and personal experience.

The attitudes are informed by various perspectives on agile software development, including: benefits, shortcomings and challenges including implementation experiences and project successes and failures, discussed in Sections 2.4.1 to 2.4.3.

2.5.1 Research question 1:

How do project managers/development team leaders describe agile software development?

2.5.2 Research question 2:

How is agile software development different to other methods of software development?

2.5.3 Research question 3:

How is agile software development similar to other software development methods?

2.5.4 Research question 4:

What are the benefits of agile software development?

2.5.5 Research question 5:

What are the shortcomings/limitations of agile software development?

2.5.6 Research question 6:

What are the challenges of implementing agile software development methods?

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

This chapter discusses the methodology employed for this research. The qualitative research paradigm is discussed together with the reasons for adopting this paradigm. The mechanisms for obtaining the data are explained and justified including the use of qualitative interviews, the sampling method, the research instrument and the way data was collected for the research. Finally the method for analysing and interpreting the data is described.

3.2 Qualitative Research Paradigm

The qualitative research paradigm was applied for the in-depth review of the perceptions of agile software development, from the perspective of managers in software development organisations.

According to Hoepfl (1997) the aims of qualitative research differ from those of quantitative research, the former seeking enlightenment, understanding and extrapolation to similar circumstances, compared to the latter which emphasises causal determination, prediction and generalisability of findings. Hazzan (2007) describes this difference in objective further by explaining that qualitative research aims to build a theoretical framework that emerges from the analysis of the research data, rather than aiming to accept or reject a hypothesis that has been defined beforehand.

The qualitative research approach allows for exploration and understanding of people-centred situations (Myers 1997; Hazzan 2007) or social phenomena, i.e. situations where people are involved and various kinds of processes are taking place (Hazzan 2007). Qualitative research is applicable in the domains of Software Engineering (Hazzan 2007) and Information Systems where there has been a shift from a technological focus to an emphasis on managerial and organisational issues (Myers 1997).

Qualitative research is appropriate where the research aims to answer questions about complex phenomena (Leedy and Ormrod 2005), particularly for exploring research questions that ask 'what' or 'how' (Creswell 1998). It is also useful where the area being researched is not that well understood (Westbrook 1994), where the topic needs to be explored and where a detailed view of the topic should be presented (Creswell 1998).

The purpose of qualitative research is to understand phenomena from the perspective of the individuals involved (Hoepfl 1997; Leedy and Ormrod 2005). It uses the natural setting as the source of data and produces a descriptive research report (Hoepfl 1997).

The objective of the research was to gain a greater understanding of the perceptions of agile methods in organisations from the software management practitioners' perspective. The qualitative research paradigm was appropriate to serve this objective in that it allowed the researcher to gain deeper insights and meaning into an area that has not been extensively researched in South Africa and from a starting point where the researcher sought to explore rather than confirm any preconceived hypotheses.

The investigation was focused on a few practitioners in the software development arena who were aware of agile development concepts and were willing to share their insights with the researcher. This aim is consistent with Leedy and Ormrod's (2005) statement that data collection for qualitative research typically involves fewer participants than quantitative research, includes those who are in the best position to shed light on a specific topic and that the research focus for qualitative research involves in-depth study.

3.3 Research Design

The qualitative interview is a powerful research tool for gathering data and is widely used in Software Engineering or Information Systems research (Hove and Anda 2005; Hazzan 2007; Myers and Newman 2007). Interviews can provide insight into people's opinions, thoughts and feelings (Hove and Anda 2005).

The interview was deemed an appropriate data collection tool for this study as it allowed the researcher to obtain multiple perspectives through conducting interviews with people from different organisations. Since the research focused on perceptions, interviews were deemed a suitable means to elicit these perceptions. The interview format allowed for details to be probed in order for the researcher to obtain a rich and deep understanding.

The data was collected using in-depth semi-structured interviews. Interviews that are used in qualitative research are typically not structured (Leedy and Ormrod 2005) and the semi-structured qualitative interview is commonly used in business and management research (Myers 2008). The semi-structured interview includes pre-formulated questions but also allows for improvisation during the interview (Myers and Newman 2007; Myers 2008) as the researcher can probe, clarify or create new questions based on what is said during the interview (Westbrook 1994). A positive aspect of the semi-structured interview is that the starting point of the pre-formulated questions provides some consistency across interviews and some focus, but the interviewee is still given the opportunity to add insights of interest as they come up during the interview (Myers 2008).

Semi-structured interviews can include specific questions to elicit anticipated information and open-ended questions to elicit unexpected types of information (Hove and Anda 2005). The researcher used closed questions for background information. She used open-ended questions to address the research questions in the interview, suggested as appropriate for qualitative research by Hoepfl (1997) and Patton (2002). According to Patton (2002), the open-ended interview allows a respondent to explain what is meaningful and significant to him/her without being “pigeon-holed” into standardised categories.

Myers and Newman (2007) warn that interviewees may withhold information if they do not trust the interviewer. Also a limited time might lead to incomplete data gathering or place subjects under pressure to the point that they create opinions not previously held before which could provide data that might not be reliable. They also note that words that are used may be ambiguous and it may not always be clear whether participants understand the questions.

Myers and Newman (2007) suggest a number of guidelines for conducting qualitative interviews aimed at minimising potential problems that can be encountered. The researcher attempted to use some of these recommendations as outlined in the table below.

Table 3: Use of guidelines for effective interviews

Guideline	Used	Method of use
Situate the researcher	Yes	The researcher explained her background and experience both before and during the interview.
Minimise social dissonance (discomfort)	Yes	The researcher dressed in business attire as all participants were from medium to large organisations, as suggested by Hove and Anda (2005). The researcher also conducted an extensive literature review to ensure that she was familiar with and able to use appropriate jargon, as per Hove and Anda's (2005) recommendation that the researcher should have extensive knowledge of the interview subject.
Represent different voices	No	The research focuses on the management perspective hence it was not appropriate to interview participants from various organisational levels.
Recognise subjects as interpreters	Yes	The interpretations of subjects themselves were captured during the interviews and texts of the interview transcripts were created and read by the researcher.
Mirror questions and answers	Yes	The researcher used open-ended questions as well as listening and prompting. The researcher was attentive, displayed interest and asked follow-up questions as suggested by Hove and Anda (2005).
Be flexible	Yes	The researcher explored lines of research as they arose; hence certain ideas were explored in depth with some participants and not with others.
Keep disclosures confidential	Yes	The researcher undertook to not reveal participant names, organisations or projects when quoting or discussing examples. Transcripts were only shared between the researcher and the interviewee.

3.4 Population and sample

3.4.1 Population

The population for this study was all project managers and development team leaders in software development organisations in South Africa. Software development organisations included those who develop software for internal use or for external clients. The participants had to be familiar with the concept of agile software development, through experience and/or research, and have been involved in a management role within a software development organisation.

3.4.2 Sample and sampling method

The sample for this study was purposive, i.e. chosen for a particular objective or purpose in mind (Pirow 1993; Leedy and Ormrod 2005; Trochim 2006). Qualitative research typically employs a relatively small and purposive sample (Hoepfl 1997; Patton 2002), i.e. chosen "*purposefully* to permit inquiry into and understanding of a phenomenon *in depth.*" (Patton; 2002:46). The logic and power of purposeful sampling comes from the emphasis on obtaining an in-depth understanding (Patton 2002). The sample in qualitative research is usually non-random (Pirow 1993).

A shortcoming of the purposive sampling method is that where the researcher wishes to extrapolate the results to a wider audience reasons would need to be given to justify this (Pirow 1993).

Patton (2002) mentions that limitations of sampling methods used in qualitative research can arise due to limited situations that are examined, changes over time and the selectivity of participants for interviews.

The researcher sought to combat potential limitations through the following actions: selecting participants that represent multiple organisations; conducting a single interview with each participant; and asking in-depth questions and

probing for sufficient detail from respondents to support interpretation and conclusions being derived.

The research made use of a specific type of purposive sampling, i.e. snowball sampling, also known as chain sampling. Snowball sampling begins by identifying someone who meets the criteria for inclusion in the research and asking them to recommend others who also meet the criteria. It is useful in accessing populations that may be inaccessible or hard to find (Trochim 2006).

Participants were selected based on their stated response that they were familiar with agile software development and on their willingness to participate in the research.

Since the researcher herself comes from a software development background and is employed at an organisation where software development is undertaken, initial participants for inclusion in the study were selected from her own organisation and professional network. Remaining participants were selected using snowball sampling.

Fifteen to twenty in-depth interviews across eight to ten organisations were planned for the purposes of gathering the data. Initial analysis of the data took place in parallel with the data collection activity with a view towards influencing the data collection. Hence results and emergent themes were monitored for convergence and a decision was taken to stop after fourteen interviews.

The fourteen respondents came from ten organisations and are listed below. One of the respondents chose to remain anonymous.

Table 4: List of respondents

Name	Title	Organisation
Ahmed Omarjee	Head of Software Development	Psybergate
Anncois Carstens	Senior Project Manager	Knotion
Anonymous	Project Manager	Rand Merchant Bank
Derek Verster	Project Manager	The IQ Business Group

Name	Title	Organisation
Greg Peringuey	Development Manager	PruHealth (Part of Discovery Health)
Michael Rogers	Enterprise Architect	Accenture
Mohammed Noorodien	Project Manager	Microsoft
Norman Blunden	Programme Manager	Discovery Health Systems
Peter Scheffel	Executive	BBD
Shainal Gosai	Head Self Service International	First National Bank
Trevor Genis	Bid Project Manager	Microsoft
Willy-Peter Schaub	Technology Specialist	BBD
Zanele Majeke	Lead Analyst	Psybergate
Zelda Hibber	IS Provider Relationship Manager	Rand Merchant Bank

3.5 The research instrument

The researcher created the research instrument specifically for the purpose of this research. The interview structure was based on the script proposed by Myers and Newman (2007):

- Opening – the researcher introduced herself, including her experience and background.
- Introduction – the researcher explained the purpose of the research and interview and what the research intended to achieve. At this stage the interviewee's personal details were also confirmed.
- Key questions – the researcher covered introductory questions on the respondent's environment and open-ended questions on the research topic. An interview guide was used, as suggested by Hove and Anda (2005) and Hoepfl (1997).
- Close – the researcher mentioned she would provide feedback to interviewees, asked permission for follow up and asked for recommendations on other people to interview. The researcher also thanked the interviewee (as suggested by Hove and Anda (2005)) and offered to provide a copy of the research to the interviewee.

As per Hoepfl (1997) the interview guide included a list of questions and general topics that the researcher was interested in exploring. The interview guide was refined based on feedback obtained from an initial review with one of the prospective interviewees and from a pilot interview conducted by the researcher.

Whilst the interview guide was prepared at the outset the researcher remained aware that the interview guide could be modified as indicated by Hoepfl (1997) and allowed for responsiveness, flexibility and improvisation during the interviews as per Myers and Newman (2007).

The interview guide used by the researcher is shown in APPENDIX A.

For reference purposes the researcher captured respondent details including, name, title, organisation, contact details and details of the interview (date, time and duration). Respondents were asked to indicate the roles they played on projects so that the researcher could confirm that they represented the research population with regard to their role, i.e. software managers.

The first section covered introductory questions on the organisation and participant. Respondents were asked about the type of company they worked in and the size of the software organisation. These questions assisted the researcher in assessing whether findings from the research could be applicable to different organisations.

Respondents were then asked a number of questions relating to their own exposure and knowledge of agile methods. Respondents were asked whether their organsations had adopted agile development practices. They were asked to rate their own knowledge of agile software development methods according to a scale of 'Very Limited' through to 'Very Extensive'. Participants were also questioned about their current and recent level of exposure to agile development. They were asked about specific agile development methods that they had exposure to and asked to rate their level of exposure as 'Low', 'Medium' or 'High'. The answers to these questions provided context for other answers provided in response to the research topics. They also served to confirm the respondents' eligibility for participating in this research by the

respondents' indication of some level of knowledge of agile development methods.

The second section focused on asking specific questions with respect to the two sub-problems being examined in this research. The interview questions mapped directly to the research questions under each sub-problem. For sub-problem 1 (Understanding of agile software development), respondents were first asked how they would describe agile software development. They were then questioned on the differences and similarities between agile software development methods and other approaches e.g. the waterfall method.

For sub-problem 2 (Attitude towards agile software development), respondents were first asked about benefits of using agile software development. Within this question they were specifically also asked if they saw agile software development as beneficial in the areas of project quality, cost and time. Respondents were then asked about the shortcomings of agile software development. Finally they were asked about challenges of implementing agile software development in practice. The researcher specifically explained the distinction between shortcomings and challenges, i.e. that shortcomings relate to inherent limitations of the method itself whereas challenges relate to the practical experience of implementing agile methods.

The questions were relatively broad and open-ended, allowing the participants to mention points that were of interest to them. Where it was required the interviewer probed and prompted further to elicit the required level of detail to answer the questions. The researcher used various types of probes (non-verbal cues and follow up questions) to obtain more detail as suggested by Patton (2002): detail-oriented probes ("who", "where", "what", "when", and "how"); elaboration probes (acknowledging, asking for more detail and examples) and clarification probes (asking for further explanation).

3.6 Procedure for data collection

Potential subjects were contacted telephonically and via email (using a standard covering letter). The purpose of the research was explained and a briefing was provided on the topics to be covered during the interview. The researcher confirmed with each potential subject that he/she met the criteria for inclusion in the study, i.e. each person worked in a management capacity in a software development environment and was familiar with agile software development. If the subject was unable to assist he/she was asked to recommend another potential participant. Based on the subject's agreement to participate, the researcher sent an electronic communication requesting an interview date.

All interviews took place face to face and each participant was interviewed once. Based on participant convenience interviews were held at participants' premises or the interviewer's premises in keeping with Patton's (2002) suggestion that people should be interviewed in locations and under conditions that are comfortable and familiar to them.

At the beginning of each interview the researcher confirmed that the interview could be recorded. Al interviews were recorded as suggested by Hove and Anda (2005). Participants were asked if their names and organisations could be disclosed, as per Hove and Anda's (2005) recommendation to discuss the matters of confidentiality and anonymity with interviewees. One participant chose to remain anonymous. The researcher also informed the interviewees that responses would be treated confidentially and that participant names and organisation specific data would not be linked to specific statements. The researcher also re-confirmed the purpose of the interview as suggested by Hove and Anda (2005).

The duration of the interviews varied (from 39 minutes to 105 minutes) depending on the level of detail provided in answers, the average interview duration being 66 minutes.

During each interview the researcher made some written notes on key items of interest as well as interviewer insights and thoughts. At the end of the interview participants were asked if they would like a copy of the research.

After the interviews written notes and recordings were transcribed to provide a full account of each interview. The complete interview transcript was sent via email to each interviewee to verify, as each participant had agreed during the interviews to review his/her transcript. Five of the fourteen interviewees responded with minor corrections to the interview transcripts and confirmation that the transcripts were an accurate account of the interviews that took place.

3.7 Data analysis and interpretation

Patton (2002) sees qualitative research as being oriented towards exploration, discovery and inductive logic. He describes the concept of inductive analysis as an analysis strategy in quantitative inquiry. Inductive analysis starts with specific observations and develops towards general patterns. The categories or dimensions of the analysis emerge from patterns found in the open-ended responses in the study, without determining in advance what the dimensions should be and without any prior assumptions or hypotheses.

This research made use of an inductive analytical approach for qualitative data analysis, described by Creswell (1998) as the “data analysis spiral” and illustrated below. This approach is consistent with the inductive analysis approaches described by Hoepfl (1997), Patton (2002) and Thomas (2003). A similar method of thematic analysis (without predefined themes) was adopted by Begel and Nagappan (2007) in their analysis of free-text responses to survey questions on perceived benefits and problems of agile development at Microsoft.

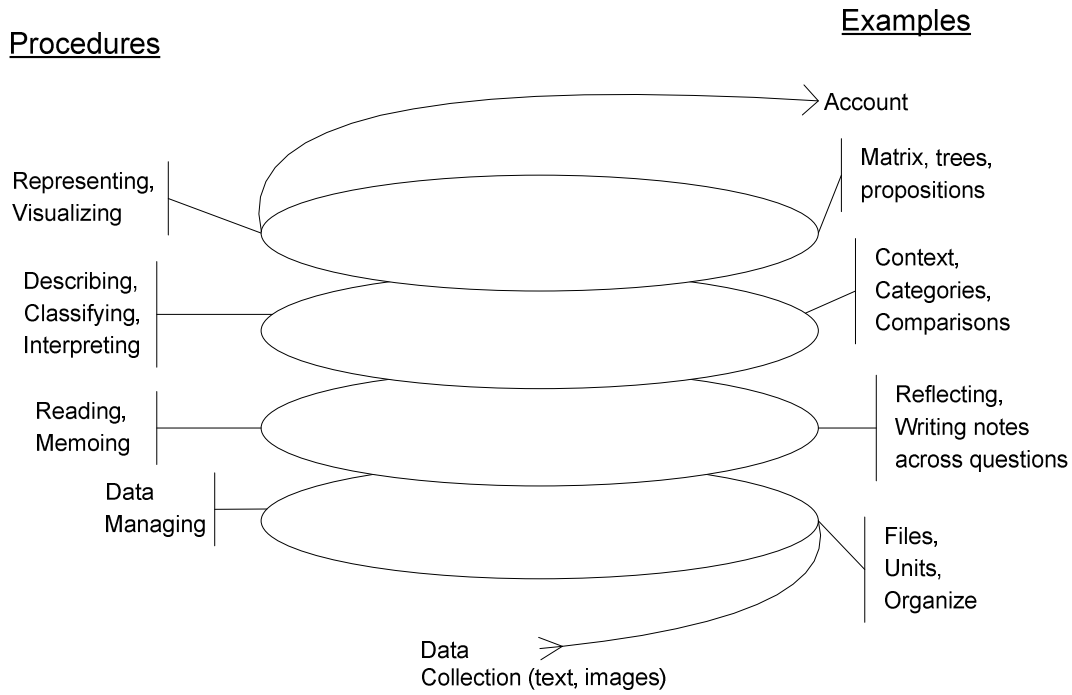

Figure 4: Data analysis spiral

Source: Creswell (1998:143)

Creswell (1998) depicts data analysis as starting with data of text or images, with the researcher following a process of moving in analytical circles and emerging with an account or narrative, i.e. managing the data, reading and annotating the data, describing and interpreting the data and finally presenting the data.

1. Managing the data

The data was organised into a spreadsheet-based computer database, as per Thomas (2003) who suggests preparing the raw data files first into a common format. Creswell (1998) recommends that the data should be organised and that the data files should be converted into smaller text units. The researcher coded each respondent's transcript data by question, respondent code and line number. This coding served as an "audit trail" scheme as indicated by Hoepfl (1997) to allow the researcher to be able identify pieces of data according to speaker and context, as participant quotes that illustrate the themes are included within the research report.

2. Reading and annotating the data

According to Creswell (1998) this step entails obtaining a sense of the data as a whole, by reading transcripts and understanding the details. Notes of key concepts can be written in the margins of field notes or transcripts. Some initial categories should be formed. The researcher perused the data and made notes of significant points covered in the data as well as potential statements that could be quoted. The researcher also initially identified where respondents had responded on more than one topic under a specific question.

3. Describing and interpreting the data

Classification involves taking the text to bits and looking for categories or themes (Creswell 1998). Words, phrases or events that appear to be similar can be grouped in the same category (Hoepfl 1997). Creswell (1998) proposes that the researcher should describe the data in detail, develop themes through some classification system and provide interpretation in light of the researcher's own views or views from literature. The researcher analysed the data (the interview transcripts) in detail. For each statement of interest a key point was identified. The key points were then categorised into themes. The researcher allowed the themes to emerge from the data rather than attempting to classify points into predefined categories. In this process some data was discarded in keeping with Creswell's (1998) suggestion that data should be screened or sifted and Thomas' (2003) view that some data may not need to be classified if it is not relevant to the research objectives.

During the detailed analysis it was evident that due to the open format of the interview respondents had sometimes provided answers relating to more than one topic in the process of answering questions. Each statement was analysed in terms of its meaning and context and where appropriate certain statements were reassigned to the relevant topic. Themes were refined a number of times. While Creswell (1998) and Thomas (2003) suggest narrowing down to a small number of themes (less than eight), the researcher chose to present all themes identified to give the reader a richer picture of the agile phenomenon. Key themes were interpreted in relation to the literature on agile methods.

4. Presenting the data

The frequency of occurrence of each theme within each topic was counted. Each occurrence was counted separately except where there were multiple occurrences within the course of single statement, where the occurrence was counted once and substantiating points were noted for inclusion in the discussion. The number of respondents who indicated each theme was also counted. For each topic a table was constructed indicating a ranking of the theme (based on frequency of occurrence), the theme, the frequency of occurrence and the number of respondents. Creswell (1998) indicates that data can be presented as text, table or figure. All themes identified are presented in the report, although only specific themes of interest are discussed in detail.

Hoepfl (1997) also considers the linkages between categories and advocates a specific step to re-examine categories to establish how they are linked. The discrete categories are compared and combined in new ways to form a 'big picture' of the phenomenon. During this process causes and effects relating to the phenomenon are identified and explored, as well as descriptive details of the phenomenon itself. As the researcher analysed and classified the data interrelationships between themes both within each research topic and across topics were identified. Tables were constructed showing the interrelationships between themes.

According to Leedy and Ormrod (2005), the qualitative inquiry is by its nature interpretive, therefore the data analysis and interpretation are closely interlinked. Both Leedy and Ormrod (2005) and Hoepfl (1997) emphasise that this process takes place repeatedly and is not linear. During the classification process the researcher revisited the original data a number of times as suggested by Patton (2002) and appropriate quotations from respondents were selected to illustrate themes of interest.

3.8 Limitations of the study

Participants for the study were selected on the basis of researcher's choice and their willingness to participate. Based on relatively small, purposive and not random sample, the sample may be considered to not be representative of all software managers in South Africa.

The data for this study was collected through face to face interviews. For reasons of practicality, time and cost for this research, the researcher was only be able to interview participants face to face in the greater Gauteng region of South Africa. However it is still expected that the results of this study would still be applicable to greater South Africa.

3.9 Validity and reliability

3.9.1 External validity

External validity relates to the issue of generalisability of conclusions drawn from the study to other situations (Leedy and Ormrod 2005). Leedy and Ormrod (2005) offer a number of recommendations to enhance external validity, which the researcher took into account during the research. The interviews took place in the work environment as per Leedy and Ormrod's (2005) suggestion to conduct research in a real-life setting. While Leedy and Ormrod (2005) suggest the use of a representative sample, the purposive sample was deemed appropriate for this research but may not be considered representative of the entire population. Replicating the study in a different context as suggested by Leedy and Ormrod (2005) was not considered to be within the scope and timeframe of this research.

With regard to qualitative research some authors (Westbrook 1994; Hoepfl 1997) look at the concept of transferability when considering external validity. Transferability of the research would depend on the extent of similarity between the original situation (i.e. the situation that was researched) and the situation to which it might be transferred to. They contend that the researcher should provide comprehensive information regarding factors that might affect

transferability and that readers themselves then can determine the extent to which findings are applicable to other situations. Stenbacka (2001) and Hazzan (2007) mention that with qualitative research, the emphasis is on careful selection of the research sample rather than the size of the sample.

The respondents were all familiar with agile software development concepts and covered a spread of organisations (size and type) in South Africa. It is likely that the research findings are applicable to most types of organisations, except potentially organisations who develop large-scale commercial or packaged software. Whilst the study included representatives from the South African consulting arm of Microsoft (a vendor of packaged software) and other representatives from product vendors, it is possible that their perspectives might not completely address the large-scale packaged software environment, which may have some peculiar challenges with regard to customer representation for instance.

Although this study focused on South Africa it is expected that most of the findings may be applicable to software organisations in other countries. A possible exception here may be on the topic of challenges of implementing agile, where some of the challenges faced in South Africa may be due to country-specific circumstances and factors such as culture and level of regulation might come into play. Practitioners in other countries should be mindful of possible differences in their countries and should interpret the findings accordingly.

3.9.2 Internal validity

Internal validity refers to “the extent to which its design and the data it yields allow the researcher to draw accurate conclusions about cause-and-effect and other relationships within the data” (Leedy and Ormrod 2005:97). Specifically for qualitative research Hoepfl (1997) and Golafshani (2003) discuss the concept of credibility in testing research findings against reality and the extent to which multiple realities that may exist in practice are represented adequately.

Leedy and Ormrod (2005) advise taking precautions to eliminate other potential explanations for the results observed. The researcher consulted with peers in the software development arena as to whether the researcher provided valid interpretations and conclusions, as suggested by Leedy and Ormrod (2005).

3.9.3 Reliability

The concept of reliability centres on repeatability or replicability of results or observations (Golafshani 2003). Reliability refers to the method or instrument's ability to consistently produce the same results (Stenbacka 2001; Leedy and Ormrod 2005). One of the forms of reliability mentioned by Leedy and Ormrod (2005) is test-retest reliability which refers to whether the same instrument being used more than once would produce the same results. As suggested by Leedy and Ormrod (2005) the researcher attempted to administer the research instrument in a consistent fashion. In order to enhance reliability the researcher tried to ensure consistency in the approach for telephone and email communications as well as for the interviews themselves. Standardised scripts, covering letters and interview guides were used.

Even with attempts to enhance reliability, it is possible that if this study were repeated the results might not be the same. The interviews consisted mostly of open-ended questions and the interviewer used prompts based on the answers given by respondents in order to elicit more information.

In the context of qualitative research some authors (Hoepfl 1997; Golafshani 2003) suggest that the important concept to be considered is dependability. Stenbacka (2001) suggests that qualitative researchers should focus on providing a thorough description of the research process and making visible all steps including preparation, data gathering and analysis. The researcher has attempted to provide a comprehensive description of her research process in the preceding sections.

CHAPTER 4: PRESENTATION OF RESULTS

4.1 Introduction

This chapter presents the results of the research. First an outline of the demographic profile of the respondents is presented, based on background questions asked about the respondents themselves and their organisations. These questions covered the following areas: primary roles; knowledge of agile software development; exposure to agile and other software development methods; situation exposure to specific agile methods; type of organisation and size of the software development organisation.

Results are then presented per research question for each sub-problem. The results of the major themes identified are presented in tables with the themes ranked per research question: description of agile methods; differences; similarities; benefits; shortcomings and challenges of implementing agile methods. Each respondent may have made multiple references relating to a specific theme. Therefore each table shows the total number of occurrences i.e. “Occurrences” (counting each reference to a particular theme) and the number of respondents per theme i.e. “Respondents” (counting each respondent who mentioned a particular theme). Finally a summary of the results is presented.

4.2 Demographic profile of respondents

The research is intended to explore the perceptions of agile software development from the perspective of software managers, i.e. project managers and development team leaders.

Respondents were asked to select one or more primary and secondary roles that they perform. The table below shows that the respondents performed multiple primary roles, all of which relate to management in a software development environment.

Table 5: Respondent primary roles

Note: Multiple responses were allowed

Role	Number
Project Manager	8
Development Team Leader	3
Architect	4
Designer	4
Developer	1
Business Analyst	2
Tester	1
Other: Account/Client/Relationship Manager, Programme Manager, Technical Researcher, Quality Assurance and Business Development	6

Considering primary and secondary roles, 13 of the 14 respondents indicated Project Manager as a role being performed. The individual who did not indicate Project Manager as a primary or secondary role was a Development Team Leader.

Respondents were asked to rate their own knowledge of agile development.

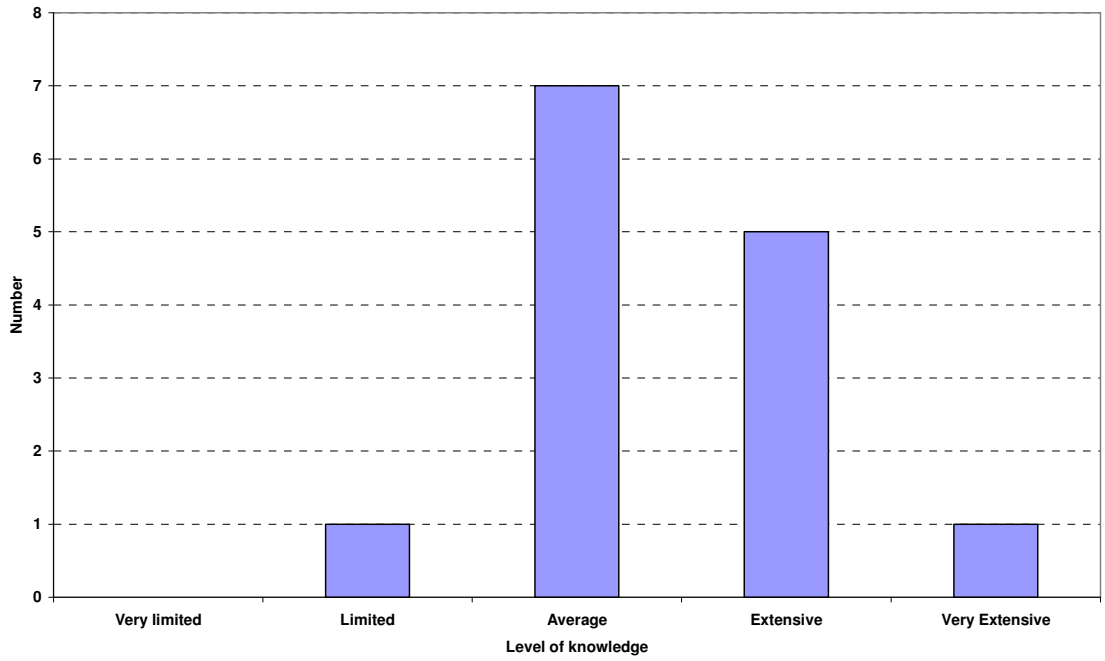

Figure 5: Respondent knowledge of agile

The graph above shows that all respondents had at minimum a 'Limited' knowledge of agile software development. Six of the respondents (43%) rated their knowledge as 'Very Extensive' or 'Extensive'.

Respondents were asked whether their organisations had adopted any agile software development practices. Of the 14 respondents 11 (79%) indicated that their organisations had adopted agile development practices.

Respondents were asked to indicate the situations in which they had exposure to agile or other software development methods. Respondents were allowed multiple responses. The table below summarises the types of situations respondents were exposed to.

Table 6: Respondent situation exposure to agile and other methods

Note: Multiple responses were allowed

Situation	Number
Currently leading Agile team	7
Considering introducing Agile for the first time	6
Currently leading a development team using other methods	8
Previously leading/member of Agile team	9

When looking specifically at exposure to agile methods, ten of the respondents were currently involved and/or previously involved in agile teams.

Respondents were asked to indicate which specific agile methods they had exposure to and rate their level of exposure as 'Low', 'Medium' or 'High'. The graph below shows that respondents had been exposed to a variety of agile methods, the most popular being XP, Scrum and Agile MSF. Nine of the fourteen respondents (64%) indicated a 'High' level of exposure to one or more of the agile methods.

Note: Multiple responses were allowed

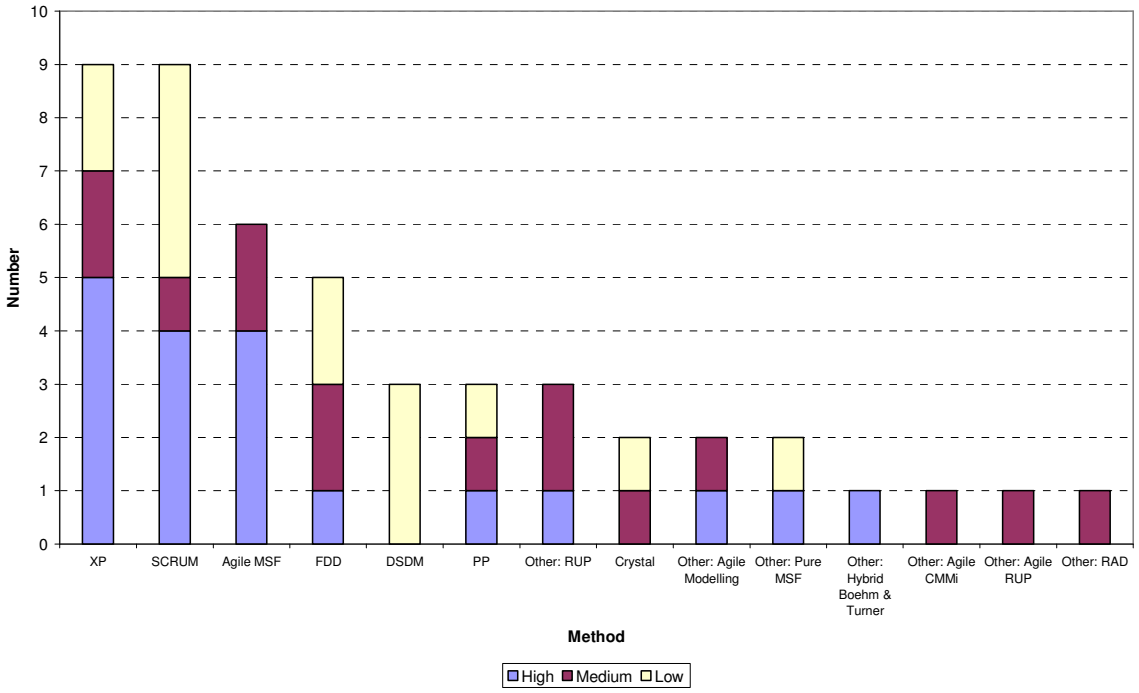

Figure 6: Respondent exposure to various agile methods

Participants were asked to indicate the type of organisation within which they were working. Respondents could choose more than one type. The table below shows the spread of types of organisations.

Table 7: Respondent organisation types

Note: Multiple responses were allowed

Type of organisation	Number
Consulting company	6
Custom/turnkey Software development company	6
Product vendor company	5
Financial services – banking	3
Financial services – healthcare	2

Nine of the fourteen respondents worked within organisations that are focused on software development, the other five represented organisations who's main line of business was not software development.

Respondents were asked to indicate the size of their company's total software organisation in South Africa, including all employees related to software development and delivery. The graph below indicates that the respondents represented companies of various sizes. The two respondents who were unable to specify the software organisation size came from organisations where the IT function is distributed across multiple business units.

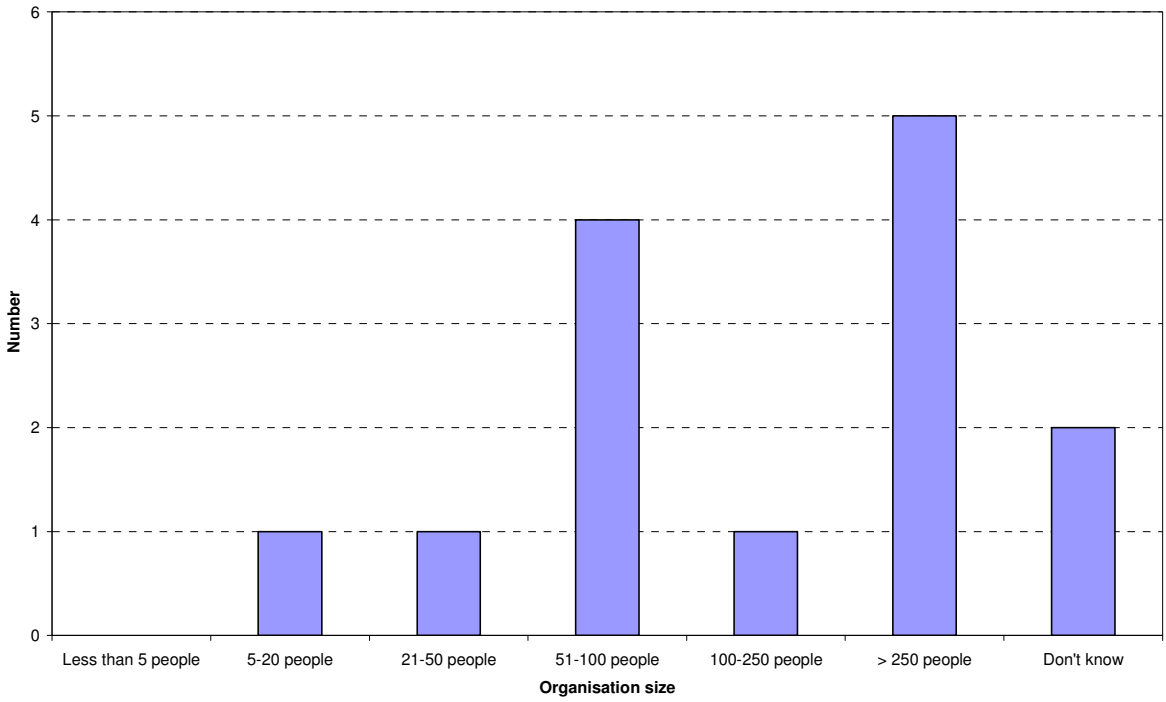

Figure 7: Respondent size of software organisation in SA

4.3 Meaning of agile software development

This section presents the themes related to each of the research questions.

4.3.1 Description of agile software development

Table 8: Characteristics of agile software development

Rank	Theme	Occurrences	Respondents
1	Iterative	23	12
2	Use agile where appropriate	23	9
3	Business is actively involved	20	11
4	Short planning and delivery cycles	18	8
5	Documentation as appropriate	14	9
6	Minimal upfront analysis/design	14	7
7	Daily/stand-up meetings	12	7
8	Small packages of work	12	6
9	Incremental/evolutionary delivery/prototyping	11	8
10	Accommodates change	10	8
11	Team mentality/ownership/responsibility	8	5
12	Comfortable visual working environment	8	3
13	Extensive developer participation	7	5
14	Team sits together	7	5
15	Communication and collaboration	7	5
16	People oriented, coaching management style	6	4
17	Pair programming/peer code review	6	4
18	Enables quicker delivery	6	4
19	Continuous/daily/automated builds	6	2
20	Aims to add value within time	5	4
21	Constant reprioritisation/revisiting work	5	4
22	Track burn rate and velocity	5	3

Rank	Theme	Occurrences	Respondents
23	Move from uncertainty/chaos to certainty	5	2
24	Trust relationship between organisation and client	4	3
25	Small teams	4	3
26	Test-driven development	4	3
27	Extensive/automated testing	3	3
29	Roles are defined	3	3
30	Requires discipline	2	2
31	The agile manifesto	2	2
32	Complements other methods	2	2
	Other: Apply agile practices, Requirements gathering with user stories, Roles are flexible/less defined, Product backlog. Define framework upfront, People and team focus, Easy to explain, Planning game, Limited bureaucracy	13	
	Grand Total	**275**	

According to participants agile development is iterative, using short delivery cycles with small work packages to deliver solutions in an incremental manner, with a low emphasis on documentation and on upfront analysis.

A common theme noted by participants is that agile methods are appropriate to certain environments or projects and that agile software development should therefore be used in situations where it is appropriate.

Respondents also noted that with agile software development the business (client) is actively involved, with a high level of trust in the relationship between the organisation supplying the solution and the customer. Agile software development accommodates changes, allowing work to be constantly reprioritised with customer feedback driving changes to the product.

Agile development is seen to be using small teams who sit together (including the client) with a high level of developer participation and communication and collaboration delivering a solution quickly to the client.

A few of the respondents mentioned the use of agile practices in general. Respondents also indicated that they used the following specific agile practices: stand-up meetings, pair programming, continuous builds, test-driven development, planning game/product backlog and user stories.

Some respondents indicated that the team roles in agile are defined but one respondent felt that they are not well defined.

4.3.2 Differences between agile software development and other software development methods

Table 9: Differences

Rank	Theme	Occurrences	Respondents
1	Documentation	28	10
2	Solution delivery	23	7
3	Dealing with changes	20	8
4	Planning/delivery cycle	19	5
5	Quality assurance/testing	16	7
6	Management role/style	16	6
7	Level of specification	16	5
8	Time/quality/scope/cost focus	14	5
9	Type of people	14	5
10	Extent of business involvement	13	8
11	Project plan	12	4
12	Project management aspects	11	6
13	Level of formality	8	4
14	Progress management	8	3
15	Specific practices	7	4
16	Process definition and flexibility	7	3
17	Lifecycle coverage	5	2
18	Team roles/members	4	2

Rank	Theme	Occurrences	Respondents
19	Developer role	4	2
20	Business analyst/developer interaction	4	2
21	Communication	4	2
22	Role definition	3	2
	Other: Customer relationship, Value for money, Contracting model, Extent of upfront estimation, Delivery time, Stakeholder identification, Level of process complexity	16	
	Grand Total	**272**	

Many of the responses indicated that there were differences between agile software development methods and other methods (e.g. traditional waterfall methods) with regard to documentation that is produced during the software development process.

Respondents noted differences in areas that were mentioned as descriptive characteristics of agile development i.e. the manner in which solutions are delivered, the planning/delivery cycle, how the different methods deal with changes, the difference in level of specification and the extent of business involvement.

There is also mention of differences in various management aspects including project management aspects, project plans, progress management and the role and style of management.

4.3.3 Similarities between agile software development and other software development methods

Table 10: Similarities

Rank	Theme	Occurrences	Respondents
1	Similar concepts, different approach	19	11
2	Deliver end product	8	5
3	Team roles/members	8	4
4	Project tracking/reporting	6	6
5	Produce documentation	5	4
6	SDLC phases	5	4
7	Project management artefacts	4	3
8	Use tools	4	2
9	Need some formality/structure/approach	4	2
10	Apply good/best practices	3	3
11	Upfront investigation/definition	3	3
12	Planning	3	3
13	Follow a process	3	2
14	Quality assurance/testing	2	2
15	Time/cost/quality constraints apply	2	2
16	Developer competency	2	2
17	Teamwork and client challenges	2	2
18	Risk management	2	2
	Other: Aim to deliver quality software, Recruiting the right people, Release management, Draw from other disciplines, Technical challenges, Iterations and builds, Development languages, Milestones	9	
	Grand Total	**94**	

Of all the questions covered there were the fewest responses regarding similarities between agile software development and other (e.g. traditional waterfall) methods.

The major theme that emerged was that agile software development has similar concepts to other methods but that the approach is different. There was extensive mention of similarities followed by 'but'; the respondents felt that the agile approach differs in terms of how things are done.

Respondents also mentioned that the end product that is delivered using an agile method is substantially similar to what might be delivered using another method.

Some respondents mentioned that the team roles/members for agile and traditional methods were similar although some respondents indicated that these were different, as shown in Table 9: Differences.

With regard to project tracking/reporting there were areas of similarity but there were differences noted with regard to progress management.

Several respondents indicated that the SDLC (system development lifecycle) phases across methods were similar. However, the planning/delivery cycle in agile methods was noted as being different.

A number of respondents noted that agile methods do produce documentation although documentation also featured highly under Differences.

A few responses mentioned that time/quality/cost constraints apply irrespective of the method, but under Differences it was noted that the focus on time, quality, scope and costs is different.

Also with regard to quality assurance and testing there were a few similarities noted but this theme also featured in Differences.

4.4 Attitude towards agile software development

4.4.1 Benefits of agile software development

Table 11: Benefits

Rank	Theme	Occurrences	Respondents
1	Quality is good/better	22	13
2	Better team spirit/dynamics	22	9
3	Meets client needs/delivers value	20	7
4	Team works better	19	10
5	Early detection of issues/failure	19	10
6	Better client involvement and relationship	18	11
7	Early/frequent delivery	17	11
8	Ability to adapt to change	16	9
9	Overall time can be quicker	15	10
10	Visibility is good	11	6
11	Cost can be lower	9	5
12	Can build initial solution and enhance	8	7
13	Less wastage	6	4
14	Promotes learning	4	3
15	Less effort	2	2
	Other: Promotes reuse, Reduces risk, Initiate more workstreams, Easy to manage, User training is easier, Facilitates outsourcing, More aligned to nature of business, Better progress management	12	
	Grand Total	**220**	

With regard to benefits respondents were specifically asked whether cost, quality and time for agile projects were seen to be benefits. Where respondents answered in the affirmative these were noted under benefits, where they indicated otherwise they were noted under shortcomings.

Almost all of the respondents felt that by using agile methods quality is good or better. Many of the responses indicated that the project time could be quicker with agile and some of them indicated that project costs could be lower.

There were many responses indicating that agile projects meet the clients' needs and deliver value to them and improved client involvement and relationships were seen as beneficial.

There were numerous responses indicating that agile methods are beneficial from a team perspective including better team spirit and improved team performance.

Many responses mentioned that with agile methods potential issues or project failures could be detected at a much earlier stage and that agile delivery is both early and frequent. Many of the responses also indicated that the ability of agile methods to cater for changes was advantageous.

4.4.2 Shortcomings/limitations of agile software development

Table 12: Shortcomings

Rank	Theme	Occurrences	Respondents
1	Limited applicability	23	7
2	Cost is not necessarily lower/can be higher	17	9
3	Definition and misunderstanding of agile	17	7
4	Less suitable for large projects/teams	11	6
5	Less suitable for distributed/outsourced/off-shore development/teams	8	6
6	Not suitable for some people	8	4
7	Can be used as an excuse	7	5
8	Less control, increased risk	7	3
9	Quality is not necessarily better/may be compromised	7	3
10	Needs hands on management	7	3

Rank	Theme	Occurrences	Respondents
11	Diminished business analyst role	6	3
12	Diminished project manager role	5	4
13	Limited upfront requirements/design	5	3
14	Difficult to provide upfront cost/time	5	3
15	Need more skilled resources	5	2
16	Knowledge sharing is more difficult	5	2
17	Time is not necessarily shorter/can be longer	4	4
18	Appears to have no process	3	3
19	Low level of adoption	3	2
20	Agile practices not always appropriate	3	2
21	Lack of traceability	2	2
	Other: Changes can have negative impacts, Can go wrong if not following agile properly, Late delivery of components can cause delays, Does not have detailed project plan, Lack of development/progress in agile methodology, Cannot run team without architecture role, Increased team stress, Can lead to never-ending projects	11	
	Grand Total	**169**	

There were numerous references to the limited applicability of agile methods (with regard to environments, projects or customers). More specifically agile is seen to not be suitable for large projects/teams and for some types of people and not working well for distributed/outsourced/off-shore development teams.

In contrast with responses that saw quality, cost and time as beneficial there were some responses with opposing views. Many of the respondents felt that agile projects did not necessarily cost less and could even cost more. A few respondents also indicated that quality might not be better or could be compromised and that time might not necessarily be shorter or could be longer.

A number of respondents mentioned factors in relation to the state of agile software development as shortcomings: agile development is itself difficult to

define and is often misunderstood, that there has been a low level of adoption of agile methods, and over the past few years there has not been much development or progress in the agile methodology itself.

Some of the respondents felt that the use of an agile method could be taken as an excuse for the team members not to do what they are supposed to do. A few responses mentioned that agile methods appear to have no process.

A few of the respondents also indicated that the diminished business analyst role and project manager role within an agile team could be problematic.

4.4.3 Challenges of implementing agile software development methods

Table 13: Challenges

Rank	Theme	Occurrences	Respondents
1	Use selectively/customise/combine agile	27	11
2	Resistance to change	25	11
3	Need education of customers and teams	14	7
4	Insufficient business involvement	13	5
5	Need the right kind of people	12	5
6	Client expectation of fixed/upfront cost/time - difficult to sell	12	4
7	Need to demonstrate/experience benefits to sell agile	11	5
8	Insufficient upfront definition	11	4
9	Need understanding of agile and coaching in order to implement	9	6
10	Integration with multiple teams/teams not using agile	9	5
11	Need agile thinking	9	3
12	Need mindset change	8	6
13	Need buy-in	8	6
14	Does not meet client governance requirements	7	4

Rank	Theme	Occurrences	Respondents
15	Tricky to measure progress and end date	7	2
16	Implementing agile disciplines/practices is difficult	6	2
17	Need to overcome misinformation/misunderstanding around agile	6	2
18	Limited tool support	6	2
19	Managing changes	3	3
20	Client not interested in method	3	2
21	Be prepared to learn along the way	2	2
22	Dealing with faster pace	2	2
23	Not applicable where client requirement/specification is clear/predefined product	2	2
	Other: Making agile work for distributed teams, Need basic formalities, Managing iterations, Less valuable where deliver on less than budget, Problem if disrupt rhythm, Need good synergy between team members, Mapping organisation team roles to agile roles	12	
	Grand Total	**224**	

A major theme that emerged from the responses relates to the way agile methods should be used i.e. that they should be used selectively, customised depending on the environment, project or team, or used in conjunction with other methods.

Many of the respondents also mentioned resistance to change as a challenge since clients and team members oppose agile methods because they are different to what they are used to. A few of the respondents noted that the parties involved in agile projects need to have agile thinking. They need to have a change in mindset and buy into agile methods in order for agile projects to be successful. Some of the respondents also indicated that there is a need to demonstrate/experience benefits in order to convince people of the benefits of agile methods.

Several respondents mentioned that there is a need for customer education on agile methods as well as a need for training/change management within project teams. Respondents also indicted that implementation of agile methods requires a thorough understanding of agile methods and there is a need for a coaching role. There is still a need to overcome misinformation or misunderstanding around agile methods, and that parties involved in agile projects need to be prepared to learn along the way.

Whilst business involvement in agile projects was seen as a key characteristic and highly beneficial, insufficient business involvement was noted to be a challenge by a number of respondents.

Some respondents indicated that where the people are not suitable it could be challenging to make an agile project work and it is also not easy to find the right people to work on agile projects.

Respondents saw as a challenge that there is a client expectation of fixed or upfront cost/time and where this cannot be provided it makes it difficult to sell agile. Insufficient upfront definition could also impact projects negatively.

A few respondents mentioned that implementing agile disciplines/practices is difficult.

4.5 Interrelationships

In examining the responses received for the topics covered by this research it became evident that the topics of characteristics, differences, similarities, benefits, shortcomings and challenges do not stand on their own and that many of the themes within these topics are interrelated.

The tables on the following pages aim to illustrate this complex web of interrelationships, by showing the major themes per topic and the relationships mentioned by the respondents.

It appears that the major descriptive characteristics of agile methods are themselves interrelated, and in turn it is these descriptive characteristics that

respondents feel are related to the differences, benefits, shortcomings and challenges of agile software development.

Table 14 below shows the relationships between characteristics, and how for example the iterative nature of agile with its short planning and delivery cycles and small packages of work allows agile projects to accommodate change. A collocated team enables active business involvement and increased communication and collaboration. Communication and collaboration also arise from the business involvement and daily stand-up meetings.

Table 14: Relationships between characteristics

Characteristic	**Related Characteristic**
Iterative	Short planning and delivery cycles
	Accommodates change
Business is actively involved	Use agile where appropriate
	Communication and collaboration
	Enables quicker delivery
Short planning and delivery cycles	Accommodates change
Minimal upfront analysis/design	Iterative
	Incremental/evolutionary delivery/prototyping
Daily/stand-up meetings	Iterative
	Accommodates change
	Team mentality/ownership/responsibility
	Communication and collaboration
	Enables quicker delivery
Small packages of work	Accommodates change
	Incremental/evolutionary delivery/prototyping
Team sits together	Business is actively involved
	Communication and collaboration
Test-driven development	Continuous/daily/automated builds

Table 15 below shows that the differences between agile software development and other methods as seen by the respondents are linked to some of the key descriptive characteristics of agile methods.

Table 15: Relationship between characteristics and differences

Characteristic	Related Difference(s)
Iterative	Planning/delivery cycle
Short planning and delivery cycles	
Business is actively involved	Extent of business involvement
Documentation as appropriate	Documentation
Minimal upfront analysis/design	Level of specification
Incremental/evolutionary delivery/prototyping	Solution delivery
Accommodates change	Dealing with changes
Test-driven development	Quality assurance/testing

As can be seen from Table 16 below the major descriptive characteristics (with the exception of "Use agile where appropriate" are seen to be related to the significant benefits of using agile methods, particularly quality, early detection of issues/failure and ability to adapt to change.

Table 16: Characteristics and related benefits

Characteristic	Related Benefit(s)
Iterative	Meets client needs/delivers value
	Early detection of issues/failure
	Early/frequent delivery
	Ability to adapt to change
	Can build initial solution and enhance
Business is actively involved	Quality is good/better
	Meets client needs/delivers value
	Early detection of issues/failure
	Better client involvement and relationship
	Visibility is good

Characteristic	Related Benefit(s)
	User training is easier
Short planning and delivery cycles	Early detection of issues/failure
	Better client involvement and relationship
	Ability to adapt to change
	Less wastage
	More aligned to nature of business
Documentation as appropriate	Less effort
Daily/stand-up meetings	Better team spirit/dynamics
	Early detection of issues/failure
	Visibility is good
Small packages of work	Early detection of issues/failure
	Ability to adapt to change
Incremental/evolutionary delivery/prototyping	Can build initial solution and enhance
Accommodates change	Quality is good/better
	Ability to adapt to change
	Cost can be lower
	More aligned to nature of business
Team sits together	Better client involvement and relationship
Communication and collaboration	Quality is good/better
	Better team spirit/dynamics
	Meets client needs/delivers value
Pair programming/peer code review	Quality is good/better
	Better team spirit/dynamics
	Promotes learning
Enables quicker delivery	Early/frequent delivery
Continuous/daily/automated builds	Quality is good/better
	Early detection of issues/failure
	Ability to adapt to change
Test-driven development	Quality is good/better
	Ability to adapt to change

Respondents also mentioned interplay between the perceived benefits of agile methods, as shown in Table 17 below. A number of the other benefits are seen to influence quality, cost and teamwork.

Table 17: Relationship between benefits

Benefit	Related benefit(s)
Better team spirit/dynamics	Quality is good/better
Meets client needs/delivers value	Better team spirit/dynamics
	Team works better
Early detection of issues/failure	Ability to adapt to change
	Visibility is good
Early/frequent delivery	Quality is good/better
	Early detection of issues/failure
	Visibility is good
Ability to adapt to change	Quality is good/better
	Cost can be lower
Overall time can be quicker	Cost can be lower
Can build initial solution and enhance	Quality is good/better
	Cost can be lower

Table 18 below shows that some of the descriptive characteristics of agile methods that were noted as beneficial by respondents are also seen to be related to shortcomings of agile methods. A number of these characteristics relate to agile methods' limited applicability and non-suitability for larger and distributed teams.

Table 18: Characteristics and related shortcomings

Characteristic	Related shortcoming(s)
Use agile where appropriate	Limited applicability
Business is actively involved	Limited applicability
	Less suitable for distributed/outsourced/off-shore development/teams

Characteristic	Related shortcoming(s)
Documentation as appropriate	Limited applicability
	Less suitable for distributed/outsourced/off-shore development/teams
	Can be used as an excuse
	Knowledge sharing is more difficult
Minimal upfront analysis/design	Limited applicability
	Less suitable for distributed/outsourced/off-shore development/teams
	Quality is not necessarily better/may be compromised
	Can be used as an excuse
	Limited upfront requirements/design
	Difficult to provide upfront cost/time
Daily/stand-up meetings	Less suitable for large projects/teams
	Diminished project manager role
	Lack of traceability
	Increased team stress
Incremental/evolutionary delivery/prototyping	Limited applicability
Accommodates change	Less suitable for large projects/teams
	Cost is not necessarily lower/can be higher
	Changes can have negative impacts
	Can lead to never ending projects
Team sits together	Less suitable for distributed/outsourced/off-shore development/teams
Communication and collaboration	Less suitable for large projects/teams
	Less suitable for distributed/outsourced/off-shore development/teams
	Not suitable for some people
Pair programming/peer code review	Not suitable for some people
	Agile practices not always appropriate

Respondents also noted that while agile methods could be beneficial some of the agile characteristics themselves seem to create challenges when implementing agile methods, as shown in Table 19 below. Some of the noted shortcomings also manifest as challenges, i.e. the difficulty in providing upfront cost/time, limited upfront requirements/design and limited suitability with regards to type of people.

Table 19: Characteristics and related challenges

Characteristic	Shortcoming	Related Challenge(s)
Use agile where appropriate		Use selectively/customise/combine agile
Business is actively involved		Resistance to change
		Insufficient business involvement
Minimal upfront analysis/design	Difficult to provide upfront cost/time	Client expectation of fixed/upfront cost/time - difficult to sell
	Limited upfront requirements/design	Insufficient upfront definition
Pair programming/peer code review		Resistance to change
Test-driven development		Resistance to change
		Need mindset change
Accommodates change		Need the right kind of people
		Tricky to measure progress and end date
		Managing changes
Roles are flexible/less defined		Need the right kind of people
Communication and collaboration	Not suitable for some people	
Team sits together		Resistance to change

4.6 Summary of the results

Respondents were primarily Project Managers and/or Development Team leaders. Their knowledge of agile methods as they rated it ranged from 'Limited' to 'Very Extensive' and the majority of the respondents had current and/or previous exposure to agile projects. The most common methods that respondents had exposure to were XP, Scrum and Agile MSF. Respondents represented a variety of organisations including internal IT departments and external IT providers within different sized software development organisations.

Regarding Sub-problem 1 (Meaning of agile software development), respondents replied to questions on the description of agile methods, differences and similarities. There was extensive response on the description (275 occurrences) and differences (272 occurrences with considerably fewer responses regarding similarities (94 occurrences).

The top five themes regarding the description of agile development were: iterative cycles; using agile methods where appropriate; active business involvement; short planning and delivery cycles and using an appropriate level of documentation.

A number of the descriptive characteristics of agile methods were related to each other and four of the top five characteristics were seen to be related to the key differences between agile and other software development methods.

Regarding differences between agile and other software development methods the top themes that emerged were: documentation; solution delivery; dealing with changes; planning/delivery cycle and quality assurance/testing.

On the topic of similarities between agile and other software development methods the most common themes that surfaced were: similar concepts but different approach; delivering the end product; team roles/members; project tracking/reporting and producing documentation.

Regarding Sub-problem 2 (Attitude towards agile software development), respondents replied to questions on the benefits and shortcomings of agile methods and the challenges of implementing agile methods. There were numerous references to benefits (220 occurrences) and challenges (224 occurrences) and relatively fewer responses regarding shortcomings (169 occurrences).

With regard to the benefits of using agile methods the top themes that emerged were: quality is good/better; better team spirit/dynamics; meeting client needs/delivering value; improved team performance and early detection of issues/failure.

Of the top themes, a number of characteristics (including four of the top five) were seen to be related to the benefits. The top five benefits themselves were inter-related.

Regarding the shortcomings of agile methods the most common themes that surfaced were: limited applicability; cost not necessarily lower and possibly higher; lack of common definition and understanding of agile; lower suitability for large projects/teams and lower suitability for distributed/outsourced/off-shore development teams.

A number of the characteristics of agile were seen to be related to the shortcomings of agile methods.

The top five themes on the topic of challenges of agile implementation were: selective, customised or combined use of agile methods; resistance to change; the need for education of customers and teams; insufficient business involvement and the need for the right kind of people.

Some of the characteristics and shortcomings of agile methods were perceived to be related to the challenges of implementing agile methods.

CHAPTER 5: DISCUSSION OF THE RESULTS

5.1 Introduction

This chapter discusses the results for each research question following on from the presentation of results. The major themes identified per research question are discussed and interpreted in relation to the literature. The research revealed a number of interrelationships between themes across the research questions and these are discussed briefly. Conclusions are drawn per sub-problem and presented at the end.

5.2 Demographic profile of respondents

This study focused on software managers i.e. project managers and development team leaders in software development organisations in South Africa, who were familiar with the concept of agile software development. Software development organisations included those who developed software for internal use or for external clients.

The respondents in this study all played a software management role indicated as Project Manager or Development Team Leader in a primary or secondary capacity.

All of the respondents were familiar with the concept of agile software development, as indicated by their knowledge of agile methods, personal exposure to agile software development in various types of situations, working within organisations that apply agile development practices and exposure to a variety of agile methods.

Respondents tended to come from various sized organisations that develop software for internal use and for external clients.

In terms of role, familiarity with agile methods and size and type of organisation, the respondents in the sample provided a fair representation of the characteristics of the population being studied.

5.3 Meaning of agile software development

This section examines the meaning of agile software development based on the research findings and within the broader context of the literature reviewed. It highlights the key descriptive characteristics of agile software methods, its differentiating characteristics in comparison to other types of software development methods and also some areas of similarity. These themes were generally consistent across respondents and organisations.

5.3.1 Description of agile software development

The descriptive characteristics discussed below feature in the top ranked themes under 'Description' and are also characteristics that displayed linkages to each other and with the other topics of benefits, shortcomings and challenges. In this section each of these characteristics is described individually.

a. Iterative

> "So the process is one of contracting that into iterations where the iterations consistently have slices of these activities in an ongoing method almost in a very rhythmical… it creates a very strong sense of momentum in the project."

Respondents agreed that a common feature of agile methods is the use of an iterative approach, i.e. cycles of planning, design, development and testing each delivering a "functional slice" to the customer. The iterative nature of agile methods is reflected as a key characteristic by numerous authors and is incorporated into methods such as Scrum, XP and MSF for Agile.

The features to be delivered are agreed at the beginning of iteration (planning game/product backlog) according to what can be accommodated within the

iteration and reviewed at the end of the iteration by the team and customer, leading into the planning session for the next cycle. Such a planning cycle is included in Scrum, XP and MSF for Agile. Of interest is the point mentioned by one of the respondents that the iterations create a sense of rhythm for the customer and bring a sense of momentum to the project, particularly where iterations run from Mondays to Fridays.

The iterative approach ties in with the agile principle of "continuous delivery of valuable software" and respondents' views are consistent with the literature.

b. Use agile where appropriate

> "…agile is an approach and it's a selective approach that's appropriate to your situation."

Respondents generally felt that agile methods are suitable for certain environments or projects. Agile methods should be used where appropriate to the extent that within an organisation there might be projects run using agile methods, traditional methods and a combination of these methods, as indicated by a number of authors.

Agile methods are seen to suit clients that are start-ups and those that are in a growth phase, who want to be first to market, who want early delivery in short cycles and who want more delivered at a lower cost. Clients should be willing to accept more risk and be willing to work with and be situated with the project team.

According to participants the method to be used varies depending on the business model or culture. One of the respondents also noted that "it's not just the process that makes an organisation agile it's also the qualities of the organisation". According to him it is unlikely that one would find a "real agile environment" in "the truly corporate environment".

The concept that there are situations that are more appropriate for agile methods is discussed extensively in the literature. The research findings are consistent with this concept although respondents focused on client and

organisational factors as being the main consideration for method selection without much mention of system and project factors that are discussed in the literature.

c. Business is actively involved

> "So where agile methods come in, they presuppose that business is actively involved in the process and that business is defining actually what they require and ultimately accepting what has been developed or not."

Participants felt that agile methods provide for active involvement of the client, as noted by a number of authors. Respondents mentioned the high degree of customer engagement, responsibility and control. Clients are part of the project team, work closely with the team and may be present all the time. Business drives the priority of the work to be done.

There is constant feedback from clients as they are shown prototypes or components of the working solution. Users can indicate if the solution is on the right track or not and may also realise if their original requirements were flawed in some way.

These concepts are consistent with the agile value of "customer collaboration", the agile principle of "business people and developers must work together daily" and the literature which talks about active client involvement, being on the same team and users prioritising their requirements. The literature suggests that the regular deliveries from the short iterations create the opportunity for clients to provide the constant feedback.

d. Short planning and delivery cycles

> "The 2 week iteration is in my opinion very good a very good starting point for most agile projects…"

The planning and delivery horizon for agile projects is short. The research revealed different views on the length of iterations or cycles. In general respondents favoured an iteration duration of one to four weeks, with the

preference being two weeks. One respondent mentioned time blocks of between one to five days. It would seem that the iteration length is influenced by the organisation and type of project. One of the respondents suggested that the risk of a project should influence the iteration length, so with riskier projects the shorter iteration (one week) would be suitable and provide a quicker feedback cycle. However a longer iteration does allow a more substantial set of features to be developed and shown to the customer.

Findings from the research are consistent with the agile principle to "deliver working software frequently… with a preference to the shorter timescale". Various authors advocate short cycles varying from under a day to up to 6 weeks, but mostly two weeks.

e. Documentation as appropriate

> "Agile does not equal no documentation. Frankly our view is documentation is always on a level as required."

Respondents acknowledge that documentation is produced on agile projects. Documentation is produced as appropriate. The amount of documentation may be less and documentation that is produced may be less detailed. Documentation may be in the form of comments in the code.

Of interest was the concept that "you can actually use rough artefacts in formal documents", i.e. less formal artefacts such as pencil sketches or whiteboard designs could be incorporated into formal documents. The same respondent also put forward the idea that certain models might be used only for the purpose of understanding for the team and might not be converted into formal diagrams for the project documentation. Another respondent mentioned the practice of reverse engineering documentation, i.e. that documentation could be derived from comments in the code using tools, at the end of the project.

Agile values and principles focus on "working software over comprehensive documentation" and "face-to-face conversation" respectively. Various authors mention the preference of informal and face to face communication over documentation. However, it was clear from the respondents that documentation

is necessary and is produced as appropriate rather than necessarily minimal or not at all. While participants were not specifically questioned regarding why they saw the need for documentation, it is possible that their use of documentation is a result of learning or experience, a feature of the use of customised agile methods or hybrid methods, or potentially as a result of client/environmental requirements.

f. Minimal upfront analysis/design

> "…you're able to deliver something that can add value or benefit to the business without them actually having too much requirements."

Agile methods aim to commence delivery with a limited level of upfront specification and analysis. Requirements are defined at a high level or "business level" and "are not set in stone". Designs are done at a high level and on the whiteboard. As the project progresses the team verifies that what is being delivered is relevant to the original goals. There is not a high emphasis on estimating the costs upfront.

Agile principles emphasise technical excellence, good design and simplicity using self-organising teams. The literature also discusses simple design for the present, as indicated from XP, the use of prototypes and the practice of ongoing refactoring. These concepts reflect the incremental rather than upfront design used in agile as indicated by the respondents.

g. Daily/stand-up meetings

> "The objective of that stand-up - maximum 30 minutes depending on the size of the team: 'what have I done, what I'm about to do, what's blocking me' - that's all I want to hear."

Respondents mentioned that a common practice of agile methods is the daily meeting, also called a stand-up or scrum meeting. The purpose of this meeting is to bring all stakeholders together for a short time (ten to thirty minutes) to discuss what each team member has done and issues encountered. Some respondents also mentioned a second meeting i.e. a post scrum meeting; this

second meeting is used where the scrum meeting is restricted to participation from developers only. Other team members (e.g. analysts) only observe during the scrum meeting but have the post scrum meeting where they all can participate and discuss issues if required.

The practice of a short daily meeting is in keeping with the agile principle of face to face communication as a superior method of conveying information and the principle of business people and developers working daily together. The literature discusses daily meetings as a method for facilitating collaboration and feedback, and the daily meeting is a feature of Scrum.

h. Incremental/evolutionary delivery/prototyping

> "And my understanding is that you don't define every little item of the development that you're going to do, you would articulate what you require and then allow the developer to either build it feature by feature or evolve from an initial outline/proof of concept, whatever it may be and then grow that to what the actual requirement is."

Respondents felt that agile methods tend to build a solution feature by feature. Instead of defining all requirements upfront, a prototyping approach is adopted, showing clients small deliverables early in the project, and evolving and improving the solution throughout the project. The focus is to deliver the most valuable features first.

The incremental approach is another manifestation of the agile principles of delivering valuable software continuously and frequent delivery of working software. The literature discusses how the agile approach delivers useful sections of functionality to customers, incorporates their feedback into evolving the product, and allows customers to dynamically prioritise features.

i. Accommodates change

> "It's a very flexible style of development as in you don't expect things to stay the same. So what we try to do is find the differences as soon

as possible and make the changes as early as possible. The idea is that you should be moving from a state of great uncertainty."

Respondents felt that agile methods encourage and welcome change and are able to cater for changes "seamlessly and conveniently". Changes arising from new market conditions, new directions or new requirements are accommodated as early as possible. The flexibility to accommodate changes comes to some extent from the smaller "bite sized pieces" and short iterations used in agile, as discussed by various authors.

The concept of accommodating change is founded on the agile value of "responding to change over following a plan" and the principle of welcoming changing requirements.

5.3.2 Differences between agile software development and other software development methods

The differentiating characteristics of agile methods compared to other traditional methods are discussed individually below. The differences that are discussed include the top ranked themes under 'Differences' and also the differences that are related to some of the descriptive characteristics mentioned above.

a. Documentation

"…we deliver the bare necessity; there is more code written than actual documentation whereas in waterfall we deliver the 'forest dump' and 'this much' working software."

The emphasis on working software as opposed to documentation comes through clearly as a difference between agile and other methods. Documentation in agile methods is typically less extensive and less formal than with traditional methods.

The lower emphasis on documentation is characteristic of agile methods according to the literature. The literature also notes the difference in the way

knowledge is managed on agile projects versus traditional methods, i.e. as tacit knowledge within the team rather than explicit internal documentation.

Of interest and not mentioned in the literature is the observation by some of the respondents that with more formal methods documentation tends to be produced upfront but is not updated during the project to reflect the implemented system, i.e. it only "looks impressive". Agile documentation where reverse engineered tends to better reflect the final product.

b. Solution delivery

> "Also what I particularly like about it is that you don't wait until the end for functionality to be completed, every week we have got a weekly iteration…"

A major difference noted by respondents is that traditional methods try to specify the entire solution upfront but deliver it at the end and typically find flaws during testing at the end. Whereas in agile methods there is early delivery of pieces of functionality at the end of iterations, with a solution that starts out small and then grows into the final solution. This difference stems from agile's incremental/evolutionary delivery model as described by respondents.

This difference is substantiated by the literature that contrasts the incremental delivery of agile methods with traditional lifecycle models such as waterfall that deliver a product only at the end.

c. Dealing with changes

> "…mandatory scope creep. Now in the formal process you can only actually bring that in at the end. Whereas with agile especially Scrum you can actually put another sticky note onto your whiteboard…and in the next planning session which is actually 2 weeks later you can plan for that feature and bring it in."

Respondents pointed out that with the waterfall method the scope is set out at the beginning and deviations are managed aggressively, whereas agile projects set an initial direction and allow for changes to be incorporated during the

course of the project. Agile methods therefore allow the team to respond to changes in market forces or changes in direction during the project as opposed to at the end. This difference in the way the methods deal with changes arises from agile method's inherent characteristic of accommodating change.

This distinction goes to the very root of agile methods and the fundamental assumption put forward in the literature that agile methods presuppose that software can continuously improve based on feedback and change compared to the traditional view that systems are fully specifiable and predictable.

d. Planning/delivery cycle

> "Fundamentally I think that the iterative nature of agile is probably one of its fundamental differentiators."

> "I think it's the frequency of iterations and the shorter delivery cycles."

Respondents saw agile methods' short, frequent, iterative cycles as different from traditional methods' longer phases and releases. This difference arises from agile methods' descriptive characteristics noted by respondents, i.e. agile's short planning and delivery cycle and iterative nature. The literature also contrasts the iterative evolutionary delivery of agile with traditional phased lifecycle models such as waterfall.

The planning on agile projects is more localised and driven by features as opposed to activities, consistent with the literature that contrasts the agile feature-driven ongoing projects with the traditional activity-driven sequential project cycle.

e. Quality assurance/testing

> "In agile software development we also believe in daily quality in the process as opposed to creating a phase for quality at the end of project cycle so inherent testing, user acceptance tests embed quality at every stage in the process."

Respondents felt that quality assurance and testing differs between agile and other methods in that agile methods test continually and iteratively rather than testing a complete solution at the end. Agile methods enforce test-driven development and extensive testing throughout the process, allowing for problems or the impact of changes to be detected early on. These concepts are consistent with the literature that mentions the daily builds and iterations in agile methods versus the single testing phase at the end in traditional methods.

f. Management role/style

> "So he observes and he directs and he's pretty much like the conductor in an orchestra. … He needs to be able to understand the intonation or inflexion or whatever it is in the various musical instruments to be able to conduct effectively and uses a process of observation and subtle direction to get the orchestra to move in the direction he wants. And I think that's different, I think traditional project managers do that differently."

> "I see it more as an organiser the glue of the team rather than managing the piece of work."

Respondents saw the agile management role as being more focused on coaching and steering rather than managing. With traditional methods there is a hierarchy where management or senior members of the team can dictate decisions. Whereas with agile methods there is a team of equals with decisions being made at various levels in a consultative fashion. The agile manager needs to play a motivational leadership role in the team.

This difference in role and style is discussed in the literature referring specifically to the leadership and collaborative style and coordination and facilitation role of the manager in agile methods compared to the traditional command and control management style.

g. Level of specification

> "I think the other difference is that you can't give somebody a spec and tell them to go and deliver off it they have to work out what needs to be developed throughout the process."

Respondents noted that with traditional methods there is extensive detailed specification upfront compared to agile methods where the design starts off as a prototype or whiteboard design and then evolves. One respondent went as far as saying that with older methods "you could have an idiot working" and he would produce exactly what was wanted as everything was pre-specified in extensive detail. Another respondent mentioned that with traditional methods there would be a long lead time before development could start compared to agile methods where teams need to be able to deliver quickly.

The difference in level of specification ties back to the characteristic of minimal upfront analysis/design that respondents mentioned and is consistent with the literature that contrasts agile's ongoing evolutionary design with traditional detailed upfront specification.

h. Extent of business involvement

> "I tell you what's different is your user is involved at all times. That is in an ideal situation."

Respondents mentioned that the extent of business involvement is different between agile and other methods. With agile methods business is involved continually providing input and making detailed decisions, including decisions on whether to exclude or change features. With formal methods, business analysts liaise between the project team and the customer with the customer only being more involved during the signoff process. The literature mentions customer involvement in agile methods as being critical compared to traditional methods that do not emphasise ongoing customer involvement; also that agile projects incorporate customer feedback into the development.

With agile methods users are more empowered than with formal methods where the project manager would potentially make more of the decisions. Of interest is the point mentioned by one of the respondents that agile methods make it easier to involve the business compared to traditional methods that were complex and difficult to explain making it difficult to involve business people.

5.3.3 Similarities between agile software development and other software development methods

The top six themes under 'Similarities' are discussed below. These reflect areas of perceived similarity between agile and other methods.

a. Similar concepts, different approach

> "…it's not they are vastly different it's just that they go about things slightly differently."

A common sentiment from respondents is that agile methods are similar in many respects to other methods but the approach is just somewhat different. This view supports the literature that suggests that the concepts in agile methods are not entirely new.

The 'similar but different' notion also comes through in the number of themes that emerge as both similarities and differences i.e. the team roles/members, project tracking/reporting and progress management, the lifecycle phases, documentation, time/quality/cost constraints and quality assurance and testing.

b. Deliver end product

> "…I can't actually see a lot of similarities but both deliver the mission at the end of the day."

Respondents felt that whatever method is used the end objective of delivering a quality product is the same and the result of a software development project is likely to be the same in terms of the end product. Various authors offer a similar

view that the goals of both types of software development methods are the same.

c. Team roles/members

> "I guess the kind of role players are also quite similar in terms of having obviously business analysts, project managers that's quite similar."

Respondents noted that the generic role players in software development projects were similar in both agile and other projects, including business analysts, testers, developers and project managers. The roles of the business analyst and project manager were specifically mentioned by some of the respondents. This similarity does not seem to be specifically addressed in the literature.

In contrast a few of the respondents mentioned that the team roles in agile projects are different. One of the respondents indicated that the balance of team members on agile projects favoured more developers and fewer analysts. Another from his experience using MSF for Agile mentioned that specific roles were included that did not feature in traditional methods. Others indicated that the roles in agile projects were less well defined and more interchangeable, which is also mentioned in the literature.

d. Project tracking/reporting

> "You still have to project manage it you have timelines you have to manage your burn rate you still need to measure and report on your progress against plan…"

Respondents acknowledge that with agile and other methods there is still a need for tracking and reporting on project progress. There is a need to track the timelines and report progress to the stakeholders. Whilst this point is not explicitly highlighted in the literature as a similarity project tracking concepts such as burndown charts, product backlogs and daily meetings to check progress are mentioned.

Some of the respondents mentioned that there were differences in the way that progress is managed on agile projects. Agile projects have multiple iterations and priorities may change during a project. The focus of agile projects is on what still has to be done versus the traditional focus on what has been achieved. On agile projects the percentage complete can be more accurately assessed from the delivery of code.

e. Produce documentation

> "There's a misconception that with agile there's no documentation."

Respondents were of the view that documentation is produced for both agile and other methods. Whilst documentation was also mentioned as a difference, the difference is seen in the extensiveness of the documentation not in the absence of documentation. This view is supported by the literature indicating the importance of documentation in all methods.

Whilst respondents were of the view that agile projects do produce documentation they also indicated that there are differences in the amount, level of detail and timing of documentation.

f. SDLC phases

> "Well there is still a system development lifecycle much as they argue against it…"

Respondents acknowledge that with agile methods a system development lifecycle is still used. Specifically the phases including analysis, design, development and testing are mentioned as being important and are performed irrespective of method.

The literature makes extensive reference to the iterative nature of agile, where the phases of planning, development and testing are covered within each iteration or cycle.

Summarising the discussion on the meaning of agile development, it is evident from the research that the descriptive characteristics of agile methods are

strongly embedded in the agile values and principles. It is these very characteristics that are seen to be distinguishing factors between agile methods and other methods of software development. Many of the respondents tended to mention the differences between agile and other methods in the process of describing the agile methods. This is perhaps not surprising given that the starting point of agile methods was to address some of the shortcomings of traditional methods. The way the agile values are stated also shows a clear preference of one set of concepts (the agile ones) over another (the concepts that seemed to typify the traditional methods).

Respondents were not able to as easily identify similarities and did not elaborate much on the similarities between agile and other methods. Although they did mention some specific similarities, majority of the respondents felt that agile methods were similar (to other methods), but still different. Some of the themes emerging under similarities were also reflected under differences.

5.4 Attitude towards agile software development

Software managers' attitude towards agile software development is discussed from the perspectives of benefits and shortcomings of agile software development and also challenges relating to the implementation of agile methods. These themes were generally consistent across respondents and organisations and no discernable difference in views emerged based on respondent exposure/experience or organisation type/size.

5.4.1 Benefits of agile software development

The benefits that can arise from using agile software development are discussed individually below. This list of benefits includes the top ranked themes under 'Benefits' and some of the benefits that are closely linked to the descriptive characteristics of agile methods.

a. Quality is good/better

> "Typically what you find in agile projects or what I've found is that you normally come in on or under time, on or under budget and the quality is most definitely better than what I've seen in the formal environment. And again that comes through early detection of problems, the visibility, all those benefits I see in the agile world."

Respondents generally felt that with agile methods quality is good or better than with other methods. The high level of client involvement and feedback contributes towards improved quality.

> "However from a business perspective because quality is a point on our agenda every day, every hour, every week, every release it is embodied in the agile spirit. Every practice that we do has some level of quality associated with it."

Some of the agile practices were specifically indicated to contribute towards better quality, i.e. continuous integration and pair programming. Respondents felt that agile methods have a better focus on quality.

There is widespread confirmation in the literature of the view that agile methods facilitate improved quality. Authors draw attention to some of the specific quality inducing practices of agile methods such as pair programming, automated testing, test-driven development, continuous integration and refactoring.

b. Better team spirit/dynamics

> "So that's my opinion the real advantage of agile it creates a team it creates spirit and by nature it becomes fun."

Respondents believed that agile methods are good for project teams and contribute to having a better team spirit. The level of engagement and cohesiveness is higher, with more energy and fun. There is a higher level of team satisfaction because they see clients' needs being satisfied. The communication and collaboration and use of daily team meetings in agile

methods contribute towards improved team dynamics. The use of pairing is also seen to assist with improved team relationships.

The positive effect of agile methods on teams is substantiated by literature which refers to improved morale and satisfaction, and an increased sense of empowerment.

c. Meets client needs/delivers value

> "…I believe certainly you end up with a product that meets their expectations; even if they didn't know what those were at the start of the project it meets those needs that they have at the end…"

Respondents felt that agile methods ensure that the end products of projects meet customer expectations. Agile methods focus on the customer needs as opposed to legalities. Value delivery is strongly linked to the iterative nature of agile – every iteration aims to add value, and each iteration provides an opportunity for the customer to assess the value that has been added. Meeting client needs is also linked to the active involvement of business that characterises agile methods. Agile methods, with the focus on collaboration and the iterative approach, assist in understanding what the client wants.

Of interest was a point mentioned by one of the respondents indicating that agile methods focus on business needs: "In agile we tend to give the business what they need not what they want." This is different to other methods that deliver what clients want just because they are paying for it.

The literature cites the benefits of increased customer satisfaction arising from having their needs met, receiving value and obtaining high quality systems.

d. Team works better

> "It's a fact the delivery is always outstanding on an agile project…"

Respondents felt that the performance of agile teams is superior in a number of ways, as a result of the way agile teams work. An agile team operates as a team of peers with high levels of communication and interaction. Developers

are involved in planning their work and work both on the technical and business aspects. These factors contribute positively to increased buy-in, commitment, collaboration, proactive thinking and output. There is a greater sense of ownership due to the consultative rather than dictatorial management approach. These benefits are supported by the literature which mentions increased productivity, the positive aspects of daily meetings, increased sense of ownership and better understanding of business aspects.

e. Early detection of issues/failure

> "In the agile approach you deliver quickly something small but it looks a bit like what they've asked for and they can very quickly realise what works and what doesn't work…"

Respondents felt that agile methods allow the project team to see early on whether they are on the right track or not and to pick up issues early. Flaws in the solution can be detected and addressed early. The success or failure of the project can be evaluated early allowing a project to be called off if required.

Early detection is beneficial and results from a number of specific descriptive characteristics noted by the respondents: short iterations; early delivery; small deliverables; business involvement and feedback; communication and collaboration; daily meetings and daily builds. Early detection of problems contributes towards improved quality.

The literature indicates that short iterations, early deliveries and customer feedback allow for identification of misunderstandings and issues. While not stated explicitly as a benefit these statements align with the respondents' views on early detection.

f. Better client involvement and relationship

> "Increased user participation and user confidence I think because your iterations are short and you're able to give your client something. He's not kept in the dark for long durations and because he's seeing delivery and he's seeing tangible results he will negotiate

> or he will interact with you in a more positive light, in a more constructive light."

Respondents felt that agile methods promote active customer involvement on a daily or almost daily basis. The client is able to determine if their expectations are met and provide feedback at each iteration. The result of this close interaction is that the clients tend to be more satisfied and happy. They see the systems team in a more positive light which contributes towards a more positive relationship between parties. The client is also able to have a better appreciation of the complexities of software development faced by the team. The benefit of improved client satisfaction is discussed extensively in the literature and mention is made of customers' needs being met as customer feedback is taken into account and changes are incorporated during the project.

g. Early/frequent delivery

> "I think conceptually speaking you're supposed to arrive at the solution far quicker than the traditional methods because you're doing this iterative thing that says lets build a piece of this and then enhance it to completion."

Respondents felt that agile methods deliver outputs early and frequently. Customers are able to see something tangible early in a project, critical features are developed first allowing business value to be realised early in a project. Early delivery is linked to agile methods' characteristic iterative approach and its ability to deliver quickly. Early delivery allows for early detection of issues and failure. These views are consistent with the description from the literature of how short iterations in agile projects provide early and regular delivery to the customer. The literature also emphasises the benefits of quick releases, which promote a focus on high value items and lead to a quicker time to market.

h. Ability to adapt to change

> "There're a number of benefits I see. Number one is you're more reactive you can react to change quicker and with IT as with everything else on this planet change is certain…"

Respondents believed that change is inevitable and thus agile methods accept change. Changes arising from environmental factors, new directions, client requests for additional features or clients' changing their minds are implemented early in a project. There is early detection and visibility of the impact of changes from a technical and business perspective. The ability to adapt to change is facilitated by agile's short iterations and small deliverables, and it is this ability to incorporate change that contributes towards improved quality.

Agile methods' ability to quickly respond to change requests is covered in the literature as well as the ability to assess the impact of changes. The agility in agile software development methods is reflected in faster responses to change requests.

i. Overall time can be quicker

> "It's much shorter it cuts down on the overall time."

Respondents generally felt that agile projects could be faster, finishing on or before time and providing quicker time to market. Customer involvement and feedback can contribute towards finishing faster. One respondent also noted that egoless development assists with the time. Another noted that agile methods allow one to easily add resources for short periods of time where required instead of increasing the overall project time. The literature focuses on improved time to market due to shorter delivery cycles rather than specific mention of reduction in overall time due to agile methods.

j. Can build initial solution and enhance

> "I think we can offer a value proposition that says 'Let's deliver what you need when you need it so you can get to market, you can realise value, get return on investment, then put more money into taking it forward'."

Respondents felt that agile methods' iterative, incremental approach allows one to start with a simple solution (and potentially take that to market) and then

enhance and refine that over iterations. One respondent noted that this approach could lead to a final solution that is more robust.

The literature describes incremental solution delivery as a characteristic of agile methods. Where the benefits of quick releases are discussed, authors note that features are reviewed per iteration and short iterations improve time to market.

5.4.2 Shortcomings/Limitations of agile software development

The top five themes under 'Shortcomings', i.e. limitations of agile methods themselves are discussed individually below.

a. Limited applicability

> "There is only a certain defined slice of projects that are suited to agile development."
>
> "…if you look at the world space in terms of how many software projects there are, how many of those could be done as agile projects? It would be under 30%."

Respondents felt that one of the inherent limitations of agile methods is that they are suited to a restricted set of projects. They felt that there were various types of systems that would not be well suited for agile development: large, complex systems with many dependencies, mission-critical systems or systems intended to be used for a long time. Some felt that agile methods were not suited to maintenance development; others felt that agile methods were not suitable where new or unproven technology was being used. Respondents felt that agile methods were not well suited to more regulated, auditable, corporate types of environments or for non-participative types of customers. They felt that certain types of projects required a greater degree of documentation, planning and analysis. The idea of limited applicability of agile methods is linked to respondents' indication that agile methods should be used where appropriate.

The literature discusses the limitation that agile methods do not suit large complex projects. A number of authors suggest that agile methods are

appropriate to certain situations. Consistent with some of the limitations indicated by the respondents are the ideas in the literature that agile methods are suited to smaller simpler systems with customers who are collaborative. While the literature suggests that agile methods are well-suited to projects with newer technologies and higher risk projects one of the respondents indicated that agile methods are not suitable for new technology projects because of the risk.

b. Cost is not necessarily lower/can be higher

> "I don't see agile delivering the same functionality at less cost. In general I see that agile delivers more features, it's more bang for your buck rather that you spend less."
>
> "I'm not convinced that it's cheaper."

Respondents were asked if there is a beneficial impact on cost of using agile methods. While a few respondents indicated that costs can be lower with agile methods, more of the respondents were of the view that with agile methods the cost is not necessarily lower and could in fact be higher. Factors that could contribute to increased costs with agile methods include: using a cross-functional team; initial learning curve costs; cost of rebuilding in multiple iterations (although the initial early delivery means lower initial costs); cost of discarded work; communication overhead and ongoing maintenance costs. One respondent noted that because it was difficult to commoditise agile development, costs might be higher. A number of respondents felt that agile methods could potentially deliver more at the same cost.

The literature mentions the possibility of cost savings with agile methods but does not seem to be conclusive on the matter of cost reduction as a clear benefit from the use of agile methods. It is possible that that experiences with project costs could vary, as agile projects run on iterations and hence costs could be controllable to some extent by managing the iterations.

c. Definition and misunderstanding of agile

> "So what is agile development? I think it's very difficult to just define because you'll get different definitions by everybody."

> "One of the biggest problems I've seen it is that people misunderstand what it is. There's a lack of understanding lots of people don't seem to go and do some reading on it."

Respondents felt that the lack of common definition of agile methods is problematic and that there are "many flavours" of agile which can cause confusion. The Agile Manifesto has to some extent addressed the fragmented view of agile but not completely. Furthermore the Agile Manifesto statements can also be seen as controversial and may not inspire confidence. There is a lack of understanding, awareness and expertise in agile methods and sometimes even misunderstanding of agile methods.

The literature does not seem to address the difficulty in defining agile methods as a shortcoming. It is possible that the definition and misunderstanding of agile may be more prevalent in South Africa as agile methods may not be as widely adopted and discussed as in other countries.

d. Less suitable for large projects/teams

> "So I think one of the shortcomings is that agile finds it difficult to grow to service larger teams. It's also I don't think it's intended to be a process that serves massive projects that haven't been split up into phases so I think that's probably one of its shortcomings in the commercial space."

> "It's easier to use traditional methods with large scale projects. Teams sit back think it all through, plan it, tell everybody what they've got to do and then keep them doing it."

Respondents felt that agile methods do not scale well and do not suit large projects and teams. With larger teams it is more difficult to maintain the high level of communication, involvement and personal interaction required for agile

development. As there is a large amount of change in an agile project, it becomes more difficult to assess the impact of changes with a larger team. Larger teams were thought to require a greater level of structure and formality. While larger teams could be split up, this would still require communication structures to be set up between sub-teams.

The lack of suitability of agile methods for large teams is supported by the literature which mentions the level of communication and interaction that is required for larger teams.

e. Less suitable for distributed/outsourced/off-shore development/teams

> "The other shortcomings are the difficulty in figuring out how to distribute agile to larger teams; so agile seems to work well in collocated teams, teams who live in same office. It's much tougher to work in a distributed team teams across the world whereas much more formalised processes tend to work better…"

Respondents believed that agile methods do not support distributed or outsourced teams as the personal interaction required for agile development would be lacking. With outsourced development more controls, documentation and specification would be required. These views are consistent with limitations regarding distributed teams that are discussed in the literature. However, in practice organisations are trying to use agile methods even in distributed settings with some success as two of the respondents reported, one of whom had experience with using agile methods in an outsourced off-shore arrangement. The literature also mentions such scenarios, noting that challenges can occur and suggestions are offered to overcome some of these challenges.

5.4.3 Challenges of implementing agile software development methods

The top five themes under 'Challenges', i.e. challenges of implementing agile methods in practice, are discussed individually below. These challenges are also the main ones that manifest from some of the core characteristics of agile methods.

a. Use selectively/customise/combine agile

> "I think the big challenge is to get the balance right. … So we realise that we don't want to be too away from agile but we need to have enough rigour to be able to prove that we follow best practice…"

> "Do what fits it doesn't have to be one specific method but do it quickly, get it done."

Respondents indicated that one of the challenges of agile methods is finding the right balance of agility to apply in projects. They acknowledged that agile methods are not applied exclusively. Respondents suggested that teams should adopt features of agile that they feel are useful and change the method to suit them. Teams can use a mix of methodologies or customise agile methods to achieve the right balance. An example of such an adaptation is conducting more upfront analysis on projects.

The literature speaks extensively about situations that are appropriate for agile development and a number of authors suggest specific approaches for selecting appropriate method(s) depending on the situation. The concept of tailoring the methodology is consistent with the literature and a number of authors advocate customising methods to suit the organisation and the team.

b. Resistance to change

> "In general customers are not really open to agile methodologies because they don't necessarily see the benefits in an agile methodology nor do they necessarily care."

> "You know I think clients have a natural suspicion about things that haven't been done…"

Respondents felt that a major challenge of implementing agile methods is dealing with resistance to change from customers and teams. Agile methods are somewhat different to what people are used to, particularly those who are used to traditional methods. People may not find agile easy to understand and customers may not really see the benefits of using agile methods.

Some of the agile practices and features such as pair programming, test-driven development, active customer involvement and the agile working environment may seem strange and clients and teams may be reluctant to adopt agile methods. Existing ways may seem proven and comfortable and people may not see how agile methods could fit into existing project management models.

Resistance to change is a challenge that is discussed extensively in the literature in relation to team member and management opposition and is largely consistent with what the research reveals.

c. Need education of customers and teams

> "There's I think there's a lot of work required around education of the organisation that you are implementing it into. They need to understand what it means to them; they often have a sense of comfort in traditional methods…. they need to understand that agile is a journey that they need to come on board with and they need to understand why they're doing that."

Respondents acknowledged that there is a need for continuous education amongst clients and teams where agile methods are to be used. Education is linked to addressing other challenges of resistance to change and the need for mind-set change. Where project teams have not previously been exposed to agile methods they also need training and change management.

The need for education is also described in the literature as being an important factor for implementing agile methods. Customer education and team training in various skills necessary to practice agile methods are mentioned.

d. Insufficient business involvement

> "Now the challenge becomes when the client is not available. Which in 90% of the cases that will happen when the user is not available."

Respondents indicated that it is often difficult to convince a client of the need for full-time involvement on an agile project. Sometimes a client may not be available to participate as a full-time member of the project team. They suggest different ways of overcoming this challenge for example using the business analyst as the user representative and liaison with the client; a consequence can be that interaction may not be as frequent and there might be delays in obtaining feedback from the client. Other suggestions are having a list of client representatives that can be contacted telephonically or providing temporary assistance for clients for their day to day responsibilities so that they have time available to participate in projects. The danger of having insufficient business involvement is that the team might make decisions on the client's behalf and ultimately find that they are not delivering business value.

The literature discusses the challenge of customers' resistance to active participation in agile projects but does not really address the issue of customer non-availability. Perhaps customer non-availability is regarded as a challenge even with traditional methods and is usually compensated for by using the business analyst as the intermediary between the project team and the customer.

e. Need the right kind of people

> "There are certain people who can work in an agile environment, there are certain who are definitely never going to be able handle it."

Respondents felt that it was important to have the right kind of people working on agile projects. In general they thought that people should be younger,

optimistic and willing to accept change. Specifically architects and developers need to be experienced and creative. One respondent felt that the business analyst role would be best filled by someone with good business experience and a lesser focus on formalised approaches. Respondents felt that team members who are rigid, have poor interpersonal skills, or don't want interaction or visibility would not work well with agile methods. Finding the right people who can work on agile methods is a challenge as in many cases the same people still need to be able to work on waterfall type of projects as well.

The literature discusses that developers using agile methods need to be highly competent and require more interpersonal skills, soft skills, technical skills and skills to use tools.

Summing up the position on agile methods, respondents generally have a favourable attitude towards agile software development. Opinions showed greater convergence regarding benefits than shortcomings. Respondents were specifically asked about whether the typical project management constraints of quality, cost and time were seen as benefits. From the responses obtained, it would seem that quality improvement and time are seen to be benefits, but that cost is seen as a potential shortcoming. Hence agile methods are seen to offer improved quality and reduced time, but not reduced costs.

However, the benefits must be seen in light of the shortcomings raised – major shortcomings were around the limited applicability of agile methods for specific types of projects and environments and the lack of suitability for large projects/teams and for distributed/outsourced/off-shore project teams. Based on these limitations it would seem that agile methods can potentially only be beneficial if used in specific circumstances.

Challenges raised were not stated as insurmountable: resistance to change, education and finding the right people could be addressed with the appropriate interventions. Teams implementing agile methods should look out for insufficient business involvement and actively manage this. The major challenge raised was in the use of agile methods – that they should be used selectively, customised and potentially combined with other methods.

It seems that the attitude towards agile software development is favourable and agile methods appear to work in practice. However, agile methods are not seen as a universal solution for all software development projects and environments.

5.5 Interrelationships

The interplay between the various characteristics of agile methods noted by respondents such as the iterative approach, the incremental approach, short cycles, business involvement, accommodating changes and delivering business value, is supported by the literature.

The key characteristics of agile methods were noted by respondents to be closely linked to the differences between agile and other methods. This consistency came through in many of the interviews themselves where respondents started describing these differences as they were describing agile methods.

Respondents felt that a number of agile characteristics were beneficial. The literature supports the concept that agile practices such as pair programming, test-driven development, continuous integration and daily meetings are beneficial. The linkages between characteristics such as accommodating changes, early delivery and business involvement to benefits of using agile methods were mentioned by respondents and are discussed in the literature. The perceived interaction between benefits such as team spirit, delivering value, early detection, early delivery and adapting to change with outcomes such as improved teamwork, quality, cost and visibility was noted by respondents and some of these linkages emerged from the literature as well.

While respondents saw that a number of the agile characteristics were beneficial, they viewed some of these same characteristics as being related to shortcomings of agile methods. Aspects such as documentation, minimal upfront analysis, communication and collaboration, team collocation, daily meetings and pair programming were seen to be related to agile methods' limited applicability to certain types of projects and clients and it's unsuitability for large and distributed teams.

Respondents also viewed some of the agile characteristics as creating challenges for implementing agile methods, particularly in terms of resistance to change and needing the right people. Also some of the shortcomings such as limited upfront costing, insufficient upfront analysis and suitability for certain types of people were seen to manifest as challenges for agile implementation.

5.6 Conclusion

Agile software development is characterised by and distinguished from other software development methods by its short iterative incremental cycles with small deliverables, execution with minimal upfront analysis and design, an appropriate level of documentation and active business involvement driving and prioritising changes through the project. Test-driven development is one of the distinguishing agile practices but other key practices include pair programming, continuous integration and daily meetings. Agile teams tend to be located together and communicate and collaborate extensively. Compared to other software development methods agile methods share similar concepts but the approach that is used is rather different. Agile methods come with a major proviso, that these methods should be used in situations (projects, environments, clients) that are deemed to be appropriate.

Agile methods are viewed positively and the factors that characterise agile methods result in a number of benefits. Agile projects deliver high quality outcomes that meet clients' needs and deliver business value. Active business involvement incorporating change, communication and collaboration and agile practices such as pair programming, continuous integration and test-driven development contribute towards improved quality. Early frequent delivery of functional software, agile methods' ability to deal with change and early detection of issues are very beneficial. Agile methods have a positive impact on team in terms of better spirit and outcome, as well as on the relationship with the client. There is also some positive impact on project costs and time.

While agile methods have advantages these should be viewed in relation to some of the shortcomings and potential challenges of implementing agile

methods. Major shortcomings relate to limited applicability in terms of certain types of projects (e.g. larger, more complex, mission-critical systems) and the requirement for certain kinds of people to make it work, i.e. those that are able to deal with change, play multiple roles and work in a very close team setting. Agile methods are not that suitable for larger team or distributed team scenarios, thereby further limiting the situations in which agile methods could be applied. The requirement of business involvement in a collocated high communication environment, potentially lesser emphasis on documentation, less upfront documentation and agile-specific practices such as pair programming and daily meetings are factors that limit agile methods' applicability. Whilst sometimes there might be benefits in terms of cost, it is more likely that project costs will be similar as when using other methods or even higher.

Amongst business people and software practitioners there is still a general lack of understanding of agile methods and even some misunderstanding on the meaning of agile. Hence there is a need for education on the subject of agile methods. Agile methods should be acknowledged as being appropriate for certain situations and inappropriate for others; hence agile methods should not be seen as the answer for every project. Instead agile methods should be viewed as another toolbox from which to select practices that might be useful. Project teams can create a customised approach for a specific set of circumstances rather than needing to choose a pure agile or pure traditional approach. Some of the practices and concepts of agile methods (e.g. active business involvement, pair programming, test-driven development) are different to what people are used to and in implementing agile methods there is resistance to change. There is a need to overcome misunderstanding of agile methods and to educate both customers and team members on agile methods so that they can change their mindsets and buy into agile. As agile methods tend to be more suited to specific types of people, finding the right people to work on agile projects can be a challenge.

CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS

6.1 Introduction

This chapter begins by reiterating the conclusions of the research in relation to the context of software development and published literature on the subject of agile methods. Recommendations are made for members of the software development community and business participants in software projects. Finally some suggestions for further research into agile software development methods are proposed.

6.2 Conclusions of the study

Agile software development methods are intended to address the challenge of business requirements and technology changes that arise during software development, by responding to change during development while also promoting high quality. A common framework for various agile methods emerged in 2001 with the adoption of the Agile Manifesto. Hence agile methods have become more well-known and used only in the past few years. Given the relative newness of the agile domain, the researcher questioned the understanding of this phenomenon by software practitioners in South Africa.

In examining South African software managers' perceptions of agile software development methods the researcher sought to extract and explore the meaning and attitude towards such methods. The research revealed that software managers perceive the meaning of agile software development to be strongly grounded in the values and principles of the Agile Manifesto. Agile software development is described as an iterative approach that breaks work into small packages and delivers a solution incrementally over numerous short cycles, accommodating changes throughout the process. The level of documentation that is produced depends on the project and the level of upfront analysis and design is minimal. Business representatives must participate actively in agile projects. Agile methods are seen to be suitable for specific

types of environments and projects. The characteristics of agile methods and their distinguishing features are inextricably linked. Whilst similarities can be identified at a very high level, the way agile methods work is quite different to other software development methods.

Software managers' attitude towards agile software development is largely positive, with a number of benefits being identified, improved quality being the most significant benefit. Other benefits relate to positive team effects, early and regular deliveries allowing for early detection of issues and accommodating changes, resulting in solutions that meet client needs with generally quicker project times. These benefits arise from the way agile projects work and some of the specific agile practices such as pair programming, continuous integration and test-driven development.

While the general sentiment is positive agile methods do have some inherent limitations which mainly centre on their limited applicability, non-suitability for large and distributed teams and suitability for only certain types of people. Of importance from a management perspective is that agile methods are not seen to reduce project costs (although some of the literature suggests that cost savings are possible). While quality and time are generally positively impacted sometimes they can be negatively impacted.

Agile methods being new and different face resistance to change during implementation. Customers and teams using agile methods require education to overcome resistance and misunderstanding of agile methods. Of major importance to management is the notion that agile methods should be implemented with some adaptation where required depending on the situation.

The findings of this study are generally consistent with the literature on the subject of agile software development methods.

This study included participants from a variety of organisations although the sample size of fourteen might be considered small. Areas for improvement on this research include using a larger sample size and possibly further analysis and refinement on the themes to narrow them to a smaller set.

6.3 Recommendations

Based on the findings of this study a number of recommendations can be proposed to members of the software development community and business people engaging in software development projects.

This study focused on the perceptions of software managers specifically, i.e. project managers and development team leaders. Software managers should consider which projects agile methods could be applied to as they might not be appropriate for every type of project situation. For projects that might not fit the typical mould of the ideal agile project some adaptation and creativity might be required in order to apply agile concepts and still ensure project success. For any software development project the best mix of practices should be adopted, drawing from agile and traditional methods, also taking note of the characteristics of agile methods and the potential resultant benefits or shortcomings. Where agile methods are being applied in environments and teams that have been accustomed to working with traditional methods, managers must be aware that there may be resistance to new agile methods and concepts. Managers must therefore take active steps in educating customers and teams about agile methods and ensure that agile projects are sold correctly.

Developers and other team members of software development projects need to understand that where agile methods are used, managers would expect high quality outputs. Developers particularly need to be aware that their participation on an agile project will extend beyond the technical aspects to encompass a good understanding of the business needs as well. They need to be prepared to work closely with team members in a setting with high levels of interaction and communication.

Customers of software should appreciate that business benefits can be attained from using agile methods and they should be willing to participate in such projects. However the realisation of these benefits requires an investment of time and energy in the form of active participation on a project.

Educators and trainers working in the arena of software development and software project management must recognise that they have a significant role to play in taking agile methods into the mainstream. They need to take advantage of this opportunity to educate people in agile methods. There is an opportunity to create forums where agile practitioners can share their experiences and lessons learnt. Also to offer formal training courses and potentially certifications for agile methods as these would assist in building up the level of knowledge and adding more credibility for agile methods.

Finally academics should recognise that the subject of agile methods is one that is ripe for exploration, particularly in the South African context. Some suggestions for further research are indicated below.

6.4 Suggestions for further research

This research focused on the perceptions of agile software development methods from the perspective of software managers in South Africa.

Similar research could be conducted amongst other stakeholders involved in software development, i.e. project team members (developers, analysts, testers, designers, architects and client representatives (users and executives), to compare and contrast the views of different stakeholder groups.

Whilst this study specifically focused on perceptions of agile methods, further more objective research could be conducted, particularly on benefits, to assess whether these perceived benefits manifest in reality and to quantify the extent of such benefits.

This study identified a number of interrelationships between the topics explored, particularly between the characteristics of agile software development and benefits. These interrelationships can form the basis for further analysis on cause-effect relationships.

This study also identified a number of shortcomings and challenges. Further research can be conducted to see how these might be addressed, building on

other research such as Ramesh et al (2006) which explores challenges in agile distributed development.

Finally a few of the respondents raised as a challenge the difficulty of selling agile methods to clients who have an expectation of fixed/upfront cost and time for projects. Further research can be conducted into sales, commercial and contractual models that are suitable for agile software development.

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. (2002) *Agile software development methods. Review and analysis*, VTT Publications 478.

Abrahamsson, P., Warsta, J., Siponen, M.T. and Ronkainen, J. (2003) 'New Directions for Agile Methods: A Comparative Analysis' in *International Conference on Software Engineering (ICSE 2003)*, Portland, Oregon, USA, 3-10 May, Proceedings of the 25th International Conference on Software Engineering, pp. 244-254.

Ambler, S. (2006) Survey Says: Agile Works in Practice, *Dr. Dobbs Journal*, 3 August 2006, last accessed 9 February, 2008, from http://www.ddj.com/architect/191800169.

Ambler, S. (2007, 15 July) *Agile Software Development: Definition*, last accessed 25 February 2008, from http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm.

Ambler, S. (2008) Has Agile Peaked?, *Dr. Dobbs Journal*, 7 May 2008, last accessed 5 December, 2009, from http://www.ddj.com/architect/207600615.

Baskerville, R. and Pries-Heje, J. (2002) 'Information Systems Development @ Internet Speed: A New Paradigm In The Making!', In *European Conference on Information Systems 2002 (ECIS 2002)*, Gdańsk, Poland, 6-8 June, Proceedings of ECIS, pp. 282-291.

Baskerville, R., Ramesh, B., Levine, L., Pries-Heje, J. and Slaughter, S. (2003) Is Internet-Speed Software Development Different?, *IEEE Software*, 20(6), pp. 70-76.

Beck, K. (1999) Embracing Change with Extreme Programming, *Computer*, 32(10), pp. 70-77.

Beck, K., Beedle, M., Cockburn, A., Cunningham, W, Fowler, M., Grenning, J., Highsmith, J., Hunt, A, Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S.,

Schwaber, K., Sutherland, J., Thomas, D. and Van Bennekum, A. (2001) *Manifesto for Agile Software Development*, last accessed 15 February, 2006, from http://agilemanifesto.org.

Beck, K. and Boehm, B. (2003) *Agility through Discipline: A Debate*, Computer, 36(6), pp. 44-46.

Begel, A. and Nagappan, N. (2007) 'Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study' in *1st International Symposium on Empirical Software Engineering and Metrics (ESEM)*, Madrid, Spain, 20-21 September, last accessed 22 February, 2008, from research.microsoft.com/hip/papers/AgileDevatMS-ESEM07.pdf

Boehm, B. (2002) Get ready for agile methods, with care, *Computer*, 35(1), pp. 64-69.

Boehm, B. and Turner, R. (2005) Management Challenges to Implementing Agile Processes in Traditional Development Organizations, *IEEE Software*, 22(5), pp. 30-39.

Ceschi, M., Sillitti, A., Succi, G. and De Panfilis, S. (2005) Project management in plan-based and agile companies, *IEEE Software*, 22(3), pp. 21-27**.**

Chow, T. and Cao, D-B. (2008) A survey study of critical success factors in agile software projects, *The Journal of Systems and Software*, 81(6), pp. 961-971.

Cockburn, A. (2000) Selecting a Project's Methodology, *IEEE Software*, 17(4), pp. 64-71.

Cockburn, A. (2002A) Agile Software Development Joins the "Would-Be" Crowd, *Cutter IT Journal*, 15(1), pp. 6-12.

Cockburn, A. (2002B) Learning From Agile Software Development – Part One, *CrossTalk The Journal of Defense Software Engineering*, 15(10), pp. 10-13.

Cockburn, A. and Highsmith, J. (2001) Agile Software Development: The People Factor, *Computer*, 34(11), pp. 131-133.

Cohn, M. (2005) *The Scrum Development Process*, last accessed 4 April, 2006, from http://www.mountaingoatsoftware.com/scrum/.

Cohn, M. and Ford, D. (2003) Introducing an Agile Process to an Organization, *Computer*, 36(6), pp. 74-78.

Conboy, K. and 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*, Newport Beach, California, 5 November, Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research, pp. 37-44.

Creswell, J.W. (1998) *Qualitative inquiry and research design: Choosing among five traditions*, 1st Edition, Sage Publications, Thousand Oaks, California.

Cusumano, M.A. and Yoffie, D.B. (1999) Software Development on Internet Time, *Computer,* 32(10), pp. 60-69.

Cusumano, M.A. and Selby, R.W. (1997) How Microsoft Builds Software, *Communications of the ACM*, 40(6), pp. 53-61.

DeMarco, T. and Boehm, B. (2002) The agile methods fray, *Computer,* 35(6), pp. 90-92.

Dyba, T. and Dingsøyr, T. (2008) Empirical studies of agile software development: A systematic review, *Information and Software Technology*, 50(9), pp. 833-859.

Erickson, J., Lyytinen, K. and Siau, K. (2005) Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research (Research Review), *Journal of Database Management*, 16(4), pp. 88-100.

Ferreira, C. and Cohen, J. (2008) Agile systems development and stakeholder satisfaction: a South African empirical study, In *2008 Annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of*

technology, Wilderness, South Africa, 6-8 October, ACM International Conference Proceeding Series, 338, pp. 48-55.

Fowler, M. (2005, 13 December) *The New Methodology*, last accessed 12 February, 2008, from http://www.martinfowler.com/articles/newMethodology.html.

Gerber, A., Van der Merwe, A. and Alberts, R. (2007) 'Practical Implications of Rapid Development Methodologies', In *Computer Science & IT Education Conference*, Republic of Mauritius, 16-18 November, Proceedings of the 2007 Computer Science and IT Education Conference, pp. 233-245, last accessed 26 November, 2009, from http://csited.org/2007/73GerbCSITEd.pdf.

Glass, R.L. (2001) Agile Versus Traditional: Make Love, Not War!, *Cutter IT Journal*, 14(12), pp. 12-18.

Golafshani, N. (2003) Understanding Reliability and Validity in Qualitative Research, *The Qualitative Report*, 8(4), pp. 597 -607.

Grossman, F., Bergin, J., Leip, D., Merritt, S. and Gotel, O. (2004) 'One XP experience: introducing agile (XP) software development into a culture that is willing but not ready', In *IBM Centre for Advanced Studies Conference*, Markham, Ontario, 5-7 October, Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research, pp. 242-254.

Haynes, S.R. and Friedenberg, M. (2006) *Best Practices in Agile Software Development*, Technical Report No. 0018, College of Information Sciences and Technology, The Pennsylvania State University last accessed 17 November, 2009, from http://www.marcfriedenberg.com/research/.

Hazzan, O. (2007) Qualitative Research in Software Engineering, *Frontier Journal*, 4(6), pp. 6-16, last accessed 17 November, 2009, from http://www.hwswworld.com/journal.php.

Highsmith, J. (2002) What Is Agile Software Development?, *Crosstalk The Journal of Defense Software Engineering*, 15(10), last accessed 15 February, 2006, from http://www.stsc.hill.af.mil/crosstalk/2002/10/highsmith.html.

Highsmith, J. and Cockburn, A. (2001) Agile Software Development: The Business of Innovation, *Computer*, 34(9), pp. 120-122.

Hilkka, M.R., Tuure, T. and Matti, R. (2005) Is Extreme Programming Just Old Wine in New Bottles: A Comparison of Two Cases, *Journal of Database Management*, 16(4), pp. 41-61.

Hoepfl, M. (1997) Choosing Qualitative Research: A Primer for Technology Education Researchers, *Journal of Technology Education*, 9(1), pp. 47-63.

Hove, S.E. and Anda, B. (2005) Experiences from Conducting Semi-structured Interviews in Empirical Software Engineering Research, In *11th IEEE International Software Metrics Symposium*, Como, Italy, 19-22 September, METRICS '05 Proceedings of the 11th IEEE International Software Metrics Symposium, pp. 23-32.

Karlström, D. and Runeson, P. (2005) Combining Agile Methods with Stage-Gate Project Management, *IEEE Software*, 22(3), pp. 43-49.

Kruchten, P. (2001) Agility with the RUP, *Cutter IT Journal*, 14(12), pp. 27-33.

Larman, C. and Basili V.R. (2003) Iterative and Incremental Development: A Brief History, *Computer*, 36(6), pp. 47-55.

Leedy, P.D. and Ormrod, J.E. (2005) *Practical Research Planning and Design*, 8th Edition, Pearson Education International, New Jersey.

Lindstrom, L. and Jeffries, R. (2004) Extreme Programming and Agile Software Development Methodologies, *Information Systems Management*, 21(3), pp. 41-52.

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, T., Kiefer, D., May, J. and Kähkönen, T. (2004) Agile software development in large organizations**,** *Computer*, 37(12), pp. 26–34.

Little, T. (2005) Context-adaptive agility: managing complexity and uncertainty, *IEEE Software*, 22(3), pp. 28-35.

Lycett M., Macredie, R.D., Patel, C. and Paul, R.J. (2003) Migrating agile methods to standardized development practice, *Computer*, 36(6), pp. 79-85.

Madsen, K. (2005) 'Agility vs. stability at a successful start-up: steps to progress amidst chaos and change', In *Conference on Object Oriented Programming Systems Languages and Applications*, San Diego, California, USA, 16-20 October, Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, pp. 313-318.

Melnik, G. and Maurer, F. (2005) 'A cross-program investigation of students' perceptions of agile methods', In *International Conference on Software Engineering*, St. Louis, Missouri, USA, 15-21 May, Proceedings of the 27th International Conference on Software engineering, pp. 481-488.

Microsoft Corporation (2002) *MSF Process Model v. 3.1*, last accessed 26 October, 2009, from http://download.microsoft.com/download/A/D/E/ADEC63C9-6235-4242-B7DE-00AF0FE7BADC/MSFProcessModelv3.1.pdf.

Microsoft Corporation (2003) *Microsoft Solutions Framework version 3.0 Overview*, Microsoft Corporation.

Miller, G. (2005) Agile Software Development for the Entire Project, *Crosstalk The Journal of Defense Software Engineering*, 18(12), pp. 9-12, last accessed 26 October, 2009, from http://www.stsc.hill.af.mil/crosstalk/2005/12/0512miller.pdf.

Mnkandla, E. and Dwolatzky, B. (2007) 'Agile Methodologies Selection Toolbox', In *The Second International Conference on Software Engineering Advances (ICSEA 2007)*, Cap Esterel, French Riviera, France, 25-31 August, last accessed 26 November, 2009 from http://www-vs.informatik.uni-ulm.de/de/intra/bib/2007/ICSEA07/data/072%20Agile%20Methodologies%20Selection.pdf.

Murthi, S. (2002) Scaling agile methods Can extreme programming work for large projects? (Critical Decisions), *New Architect*, 7(10), pp. 14-15.

Myers, M.D. (1997) Qualitative Research in Information Systems, *MIS Quarterly,* 21(2), pp. 241-242, last accessed 22 March, 2006, from http://www.misq.org/discovery/MISQD_isworld/.

Myers, M.D. (2008) *Qualitative Research in Business & Management*, Sage Publications Ltd., London.

Myers, M.D. and Newman, M. (2007) The qualitative interview in IS research: Examining the craft, *Information and Organisation*, 17(1), pp. 2-26.

Nerur, S., Mahapatra, R. and Mangalaraj, G. (2005) Challenges of migrating to agile methodologies, *Communications of the ACM*, 48(5), pp. 73-78.

Newkirk, J. (2002) 'Introduction to Agile Processes and Extreme Programming', In *International Conference on Software Engineering (ICSE '02)*, Orlando, Florida, USA, 19-25 May, Proceedings of the 24th International Conference on Software Engineering, pp. 695-696.

Patton, M.Q. (2002), *Qualitative research and evaluation methods*, Third edition, Sage Publications, Thousand Oaks, California.

Pirow, P.C. (1993) *A Guide for Management Research*, Woodacres Publishers, Johannesburg.

Ramesh, B., Cao, L., Mohan, K. and Xu, P. (2006) Can Distributed Software Development be Agile?, *Communications of the ACM*, 49(10), pp. 41-46.

Rising, L. and Janoff, N.S. (2000) The Scrum software development process for small teams, *IEEE Software*, 17(4), pp. 26-31.

Schatz, B. and Abdelshafi, I. (2005) Primavera gets agile: a successful transition to agile development, *IEEE Software*, 22(3), pp. 36-42.

Stenbacka, C. (2001) Qualitative research requires quality concepts of its own, *Management Decision*, 39(7), pp. 551-556.

Sutherland, J. (2001) Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies, *Cutter IT Journal*, 14(12), pp. 5-11.

Theunissen, W.H.M. (2003) *A case-study based assessment of agile software development*, Unpublished MSc Thesis, University of Pretoria, Pretoria.

Theunissen, W.H.M., Boake, A. and Kourie, D.B. (2005) 'In search of the sweet spot: agile open collaborative corporate software development', In *2005 Annual Conference of the South African Institute of Computer Scientists and Information Technologists* (*SAICSIT 2005)*, White River, South Africa, 20-22 September, Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries, 150, pp. 268-277.

Theunissen, W.H.M., Kourie, D.G. and Watson, B.W. (2003) 'Standards and Agile Software Development', In *2003 Annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology (SAICSIT '03)*, Johannesburg, South Africa, 17-19 September, Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists, 47, pp. 178-188.

Thomas, D.R. (2003) *A general inductive approach for qualitative data analysis*, School of Population Health, University of Auckland, New Zealand, last accessed 20 February, 2008, from www.health.auckland.ac.nz/hrmas/resources/Inductive2003.pdf.

Trochim, W.K. (2006) 'Nonprobability Sampling', *Research Methods Knowledge Base*, last accessed 16 February 2008, from http://www.socialresearchmethods.net/kb/sampnon.php.

Turk, D., France, R. and Rumpe, B. (2002) 'Limitations of Agile Software Processes', In *Third International Conference on eXtreme Programming and Agile Processes in Software Engineering*, Alghero, Sardinia, Italy, 26-29 May, Proceedings of the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, pp.43-46, last accessed 22 February 2008 from http://www.agilealliance.com/system/article/file/1096/file.pdf.

Turk, D., France, R. and Rumpe, B. (2005) Assumptions Underlying Agile Software-Development Processes, *Journal of Database Management*, 16(4), pp. 62-87.

Turner, S.V. (2006) 'Chapter 1: What Is MSF and Is It Right for You?', *Microsoft ® solutions Framework essentials*, Microsoft Press, last accessed 26 October, 2009, from http://www.polyteknisk.dk/related_materials/9780735623538_chapter_01.pdf.

Van Deursen, A. (2001) Customer involvement in extreme programming: XP2001 workshop report, *ACM SIGSOFT Software Engineering Notes*, 26(6), pp. 70-73.

VersionOne Inc. (2007) 'Survey Highlights & Full Data Report', *2nd Annual Survey "The State of Agile Development"*, VersionOne Inc, Alpharetta, last accessed 9 February, 2008, from http://www.versionone.com/pdf/StateOfAgileDevelopmet2_FullDataReport.pdf.

VersionOne Inc. (2008) 'Full Data Report', 3rd *Annual Survey "The State of Agile Development"*, VersionOne Inc, Alpharetta, last accessed 22 October, 2009, from http://www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf?mkt_tok=3RkMMJWWfF9wsRonsq3fLqzsmxzEJ8357%2BwoWbHr08Yy0EZ5VunJEUWy2YMD.

Wagner, L. (2001) Extreme Requirements Engineering, *Cutter IT Journal*, 14(12), pp. 34-38.

Westbrook, L. (1994) Qualitative Research Methods: A Review of Major Stages, Data Analysis Techniques, and Quality Controls, *Library & Information Science Research*, 16(3), pp. 241-254.

Williams, L. (2004) *A Survey of Agile Development Methodologies*, last accessed 28 February, 2006, from http://agile.csc.ncsu.edu/SEMaterials/AgileMethods.pdf.

Williams, L. and Cockburn, A. (2003) Agile Software Development: It's about Feedback and Change, *Computer*, 36(6), pp. 39-43.

Young, C. (2005, 16 December) *MSF 4.0 and Microsoft Team Services*, last accessed 26 October, 2009 from http://geekswithblogs.net/cyoung/articles/63354.aspx.

APPENDIX A

Actual Research Instrument

Opening

Thank you for agreeing to participate in my MBA research.

My name is Kalpana Parshotam. I am in the process of completing my MBA at Wits Business School. My background is in Information Systems and I have been working in the IT and business consulting field for the past 10 years. I have performed various roles in both software development and consulting projects, including development, testing, business analysis and project management. I am currently employed at The IQ Business Group as a manager in the Healthcare cluster.

Introduction

As I explained briefly in our telephonic/email discussion, I am investigating the perceptions of agile software development methods by project managers/development team leaders in South Africa. You indicated that that in your role as project manager/development team leader you are familiar with the concept of agile software development.

The purpose of today's interview is for us to explore your understanding and opinions of agile software development. The intention is for me to obtain a deeper understanding of how agile software development is viewed, specifically from a development management perspective.

Our interview should not take more than 2 hours. I am going to be audio-taping our interview so that I can capture all the details of our discussion.

Would it be acceptable to you if I disclose your organisation and your name as being participant to this study? Your name/organisation will still not be linked to

any specific statements. However, if you prefer, you and your organisation can remain anonymous.

Note: *Anonymous?*_____

Respondent Details

Respondent Name:

Title:

Organisation:

Project roles: (a person may play multiple roles)

Role	**Primary role (✓ or X)**	**Secondary role (✓ or X)**
Project Manager		
Development Team Leader		
Architect		
Designer		
Developer		
Business Analyst		
Tester		
Other (specify)		

Contact Telephone Number:

Email Address:

Date of interview:

Time of interview:

Duration of Interview:

Key Questions

Section A: Introductory questions on the organisation and participant:

1. What type of company does the software development organisation you work in fit into?

Consulting company	
Custom/turnkey Software development company	
Product vendor company	
Other company whose main line of business is NOT software development (Please specify)	

2. How large is your company's total software organisation in South Africa, including all employees related to software development and delivery?

Less than 5 people	
5-20 people	
21-50 people	
51-100 people	
100-250 people	
> 250 people	

3. Has your company adopted Agile development practices anywhere within your software organisation?

Yes	
No	

4. How would you rate your knowledge of agile software development methods?

Very limited	
Limited	
Average	

Extensive	
Very Extensive	

5. Which situations below describe your current/recent level of exposure to agile development?

	Yes	No
Currently leading an Agile development team		
Considering introducing Agile within a team or organisation for the first time		
Currently leading a development team using other (Not Agile) methods		
Previously leading/member of an Agile development team		

6. Which agile development methods have you had exposure to?

XP	
SCRUM	
FDD	
DSDM	
PP	
Crystal	
Agile MSF	
Other (Specify):	

Section B: Questions relating to the research topic:

(Sub-problem 1: Understanding of agile software development)

1. How would you describe agile development?

2. How is agile software development similar to other approaches?

3. How is agile software development different to other approaches?

(Sub-problem 2: Attitude towards agile software development)

4. What do you see as positive attributes/benefits of agile software development?

5. What do you see as negative attributes/shortcomings of agile software development?

6. What do you see as the challenges of implementing agile software development methods?

   Allow participant to explain as he/she understands and only prompt/explain where required:

   Use probing techniques to elicit more information:

   Detail: who, what, where, when how

   Elaboration: acknowledge, continue, examples

   Clarification

Close

Thank you for your time and for sharing your opinions and experiences with me.

As my research progresses, I will be writing up the findings. I would like to circulate these to you for your feedback/comment on the interpretation and conclusions that have been drawn. Would you be prepared to participate?

Would you like a copy of my research once it has been completed?

(Ask only if more respondents are required. Ask this prior to the interview)

Are there any other people you know inside or outside your organisation that perform a similar role to you, are familiar with agile software development, and would be interested in participating in my research?

Name:

Organisation:

Contact Telephone Number:

Email Address: