Student: Denis the Menace
Matriculation Number: S102xxxx

Course: BSc (HONS) Computing (Information Systems Development)

Module: Honours Research and Project Methods
(MHG405297)
Module Leader: Richard Foley

Project Title: An investigation to find if a hybrid methodology, combining Scrum and Extreme Programming, can contribute to overcoming the software development project failure characteristics attributed with the use of a prescription approach.

Supervisor: Richard Foley
2nd Marker: June McCrae

HONOURS FINAL PROJECT REPORT

"Except where explicitly stated, all the work in this document is my own"

Signed_____ Date_____

Abstract

The advancement in technology has placed great pressure on software developers to produce more complex software so as to exploit the benefits that this new digital era can provide. Methodologies necessitate that a disciplined approach to software development is taken with the primary aim of making the process more predictable and therefore more efficient, consequently reducing the potential for risk to exist. The most recent approach which has been taken, with the aim of making the process of software development more risk free, is a set of methodologies referred to as Agile Methodologies. These agile methods offer a technique to overcoming the limitations found in the traditional approaches and boast greater flexibility and ability to change with the software development process.

This project focuses on two of the most widely implemented agile methods, Extreme Programming (XP) and Scrum. Each method elicits several practices which when implemented enable the method to be considered agile in its nature. As failure based statistics within software development projects remain a constant and impotent source of poor reading it is ever evident that a solution is required to alter this fact. This project begins by investigating the leading reasons responsible for the high failure rates. It then continues to analyse each of the practices within XP and Scrum to determine which are the most effective in overcoming the common failure areas, resulting in the amalgamation of practices to combine a unified method which is more effective in regards to minimising failure but continues to uphold the philosophy of Agile. The study employs qualitative and quantitative based analysis techniques such as a 4 Dimensional Analytical Tool and gap analysis to deduce data as well as a questionnaire providing supportive data funding the requirement for this project.

The results of the project highlight that both of the chosen methodologies provide their own strengths towards software development projects but there are areas which both support poorly and it was these areas which the unified method had to overcome. The research carried out illustrated that it was possible to create a unified method which combined the strengths of each to produce a more effective methodology which also provided solutions to overcoming problems areas which, before, were not effectively managed.

Acknowledgements

I would like to take this opportunity to say thank you to my supervisor, Dr Richard Foley, for all of the help, advice and feedback that he has provided me with throughout the honours year and the completion of my research project.

I would also like to further thank my first supervisor Mr Edwin Gray for his support and advice at the beginning of my project, and his valued insights into my chosen topic area.

I also extend a thank you to each of participant who took the time to complete my questionnaire which enabled me to gain valuable information relating to my project topic.

List of Table and Figures

1.0 Introduction

1.1 Background

The advances made in technology have placed great pressure on software developers to produce more complex software so as to exploit the benefits that this new digital era can provide (Boehm et al., 2009). Methodologies dictate that a necessitate approach to software development is taken, with the primary aim of making the process more predictable and therefore more efficient, consequently reducing the potential existence of risk.

The Three Approaches to Software Development

Within the software development community there are three recognised categories of approach; heavyweight, middleweight and lightweight (Guntamukkala et al., 2006). Heavyweight approaches are careful, methodical but considered slow in terms of development time. Of the heavyweight methodologies that exist, the waterfall methodology is often regarded as being the most stereotypical traditional software development methodology (Ji& Sedano, 2011). The first formal publication illustrating the waterfall methodology was in 1970 by Winston W. Royce. In this publication Royce claimed that for small projects only the completion of two development stages, analysis and coding, were required. However, Royce goes on to state that taking this approach with a larger, more complex project would have it "doomed to failure" (Winston W. Royce, 1970) due to the possibility that the initial requirements or project scope may be altered by the client which would be problematic as the waterfall method is not adaptable to change.

Middleweight approaches such as the incremental and spiral models adopt many of the same characteristics held by a heavyweight approach but they have the additional benefit of being more adaptable to change, essential in modern software development (Guntamukkala et al., 2006).

Whilst both heavyweight and middleweight approaches have key benefits in the development of software, such as being thorough and ensuring detailed documentation, they are somewhat old fashioned in the present trend of software development. However, as Guntamukkala (2006) recognises many software developers make the mistake of believing that because of their existence for such a broad period of time and their implementation in such a vast amount of projects, that they are the safest choice when adopting a methodology to put in place for a new software development project. This is part of the problem that must be overcome if the statistics of IT projects are to improve.

The final approach to software development is the lightweight approach or 'agile' approach. Agile software development methodologies aim to combat the uncertainty of modern business head on by being "free-spirited" (Hazzan& Leron, 2010; 2010)in their approach. The idea is that an agile developer reacts to the "business, technology and social environment" (Beck& Boehm, 2003) that surrounds their task intuitively to avoid inefficient product development. Since the creation of the Agile Manifesto in 2001, the software development community has seen the successful rise of methodologies such as Extreme Programming and SCRUM. The basis for their success lies with the ability to be more

flexible than traditional methodologies and having quicker development times, thus making them much more suitable to the demands of modern software projects.

The Problem

The modern software development world is a very hostile environment that continues to move forward, technologically, at an alarming pace. The success of software engineering is based on the delivery of quality software, where quality refers to the end product being delivered on time and meeting the clients requirements (Akingbehin, 2005). Research carried out by Oxford University in 2003 (Cuthbertson& Sauer, 2003) found that of the 421 projects studied, 35% were identified to finish behind schedule; 59% were found to finish over budget; and 54% of projects under delivered on initial commitments. These facts clearly highlight that the current method in which IT projects are being carried out is detrimental to the economy and is placing great strain on those involved in technology based projects. The research completed by (Cuthbertson& Sauer, 2003)) illustrates how complex and high risk a software development project can be and with modern day organisations being so dependent on the functionality provided to them by their software systems, without which their survival would not be possible (Hinde, 2005), exemplifies that a change in approach is required.

In order for a business to not only survive but to profit in this intimidating environment, the continuous task of developing innovative solutions, to reduce the existence of risk in software development, must be taken on with diligence and creativity. The identified difficulty of developing and maintaining software has become a highly problematic area in software development and the dismal statistics which accompany the problem has led to the current situation being referred to as a "crisis" (Krishnan et al., 1999). Those involved in software development must therefore come to realise that, just as technology constantly changes so must the way in which it is developed.

Whilst agile methodologies have been considered a conventional software development process for almost a decade, there is still a strong feeling of uncertainty towards them due to their unstructured personality. Highsmith (Highsmith& Cockburn, 2001) invokes the idea that the seemingly tempestuous approach has brought about much debate as to their effectiveness for software development projects with the view of some specialists being that they are too prone to risk and that the misapplication of such methods can have adverse consequences (Boehm, 2002; 2002) because they do not focus heavily enough on formal techniques, such as documentation, in the way more traditional software methodologies do such as the waterfall method.

Why Agile?

The word 'agile' is defined as being "able to move quickly and easily" (Oxford Dictionary, 2011). Goldman (Goldman et al., 1995) defines agile in its relation to business as: "Agility is dynamic, context specific, aggressively change-embracing, and growth-oriented". The word portrays an idea of dexterity as well as the impression of an inconsistency in its purpose which is the primary reason that the interest in agile methodologies has seen an increase as it mimics the hostile environment in which it is practiced.

Agile methodologies offer many benefits to software development over the more traditional development methods. One such beneficial aspect of an agile methodology is having working code to demonstrate to the client exactly what has been done, whereas a more traditional approach would usually iterate what the client will have in the future, thus making the code very palpable (Vijayasarathy& Turk, 2011). This tangible construction provides a certifiable means of measuring product development.

Agile has also been adopted due to its ability to adapt to change. A key principle of agile is that a change in the requirements is accepted with little hassle or disruption to the projects development (Greer& Hamon, 2011). This ability to welcome change ensures that even in the latter stages of the development process the client can still change the requirements of the products functionality allowing the client to constantly maintain a "competitive advantage" (Beck et al., 2001)in their respective field.

The Solution

As the debate continues between the traditionalists and the supporters of agile, there are those in the middle who argue that by combining multiple methodologies in proportionate measures, to help provide a tailored way in which to meet the clients' requirements of the project is possible (Boehm& Turner, 2003).

The difficulty with software development projects is that every project is different and so creating a methodology which can be instigated with every project is near impossible. Each project has different requirements, timescales and risks and therefore a new methodology framework cannot be created with the sole aim of covering all potential aspects of every project, the key is balance. The book Balancing Agility and Discipline: A guide for the perplexed, written by Barry Boehm and Richard Turner, describes several techniques to aid the decision of how to approach a software development project, dependant on the projects requirements. One such solution which has been implemented and found to produce positive effects is the combining of multiple methodologies, taking the practices or values of methods which apply most to the project and which can offer the greatest support for overcoming difficulties likely to face the project. Therefore the solution which is created in this report does not aim to meet the requirements of all potential project types but instead has the objective of reducing the common development failures by providing greater support enabling software development projects more low risk.

The Choice of XP and Scrum

Created by Kent Beck, XP bases itself on four fundamental values which must be upheld if success is to be achieved; Communication, Simplicity, Feedback and Courage. The concept of XP is based on that of an iterative methodology where the project is broken down into small phases, allowing the project team to work on "a little at a time" (Beck, 1999) throughout the entire software development. Ambler (2000) clarifies that XP is aimed at being used by small sized teams which operate in a hostile environment, that is to say one in which the requirements of the project can change at any time without prior warning.

Developed by Ken Schwaber and Jeff Sutherland, SCRUM is based on the concept that software development is not a defined process, but an empirical process which can witness

complex transformations between inputs and outputs that may or may not be repeated under differing circumstances. SCRUM is a high level agile software development framework process for incrementally constructing software within complex environments with the intention of being worked on by small project groups, designed to produce a tangible deliverable after thirty days development (Rising& Janoff, 2000). It is very adaptable to change and is described as being a "client centric approach" (Santos et al., 2011) because the software development team often has a representative of the client on the team at all times to reduce make sure that the development follows the initial requirements.

The reason as to why these two software development approaches have been selected for further analysis within this project is due to the fact that both have gained great support from the agile software development community. Furthermore, both are widely recognised and implemented in software development projects therefore meaning that research which could benefit either methodology would be worthwhile.

1.2 Project Outline

This research project follows the optimism that Spence (Spence, 2005) illustrates, that there must be a "better way" of finding a software development method that can be applied to a wider spectrum of projects and which provides support to overcoming the current failure areas that exist. This project will identify the most common software development project failures through a study of literature. The project will go on to analyse two agile software development methodologies to identify how they can provide solutions in overcoming the problematic areas whilst also detecting which of the recognised failure areas are poorly supported by both methodologies. The second part of this project will then attempt to combine the most effective practices from both methodologies with regards to providing support to the failures, identified through analysis, thus producing a unified methodology which is more supportive of common problem areas but which still follows the values of an agile method.

1.2.1 Research Question

Modern software projects are widely regarded as being difficult to manage due to their complexity, high risk factors and because the product development is not tangible in the same way as more traditional engineering projects. The difficulty facing project managers is finding the correct balance when adopting a methodology, ensuring it is sufficiently agile to cope with the changing environment but disciplined to ensure that the completed end product is of a high quality and meets the clients' requirements. Both XP and Scrum bring with them advantages and disadvantages, however it must be noted that modern software development characteristics invoke that a unification of the two approaches is most beneficial (Boehm, 2002). It is for this reason and the issues identified in the background research that my research question is:

"Can a hybrid agile methodology framework be constructed based of on the amalgamation of existing agile methodology principles and practices, derived from XP and Scrum, and combining with disciplined principles to construct a unified methodology which can improve support given to overcoming common project failures, evaluating the results by simulating

the implementation of the amended methodologies against the application of existing XP and Scrum methodologies, identifying the advantages through focused analysis?"

The primary reason for the selection of these two methods to be used within this research is because they are two of the most cited methodologies in current publications (Fernandes& Almeida, 2010) thus producing the second reason, that there is obviously a large interest and following of these methodologies which highlights that any research which could help develop their future implementation could be beneficial to the software development community.

1.2.2 Hypothesis

From the 2005 Agile Conference, Spence's article entitled "There has to be a better way!" (Spence, 2005) conveys that there must be a more beneficial approach to balancing agility with discipline. The amalgamation of practices from XP and Scrum, which are identified as being the most supportive towards overcoming the common failures which face software development projects, may provide a way in which to create a stronger unified methodology which could prove to be more supportive of problematic areas that neither method can support effectively in an independent situation.

1.2.3 Objectives

The primary aim of this project is to investigate the potential possibility of a new software development methodology being created through unification of the strengths of XP and SCRUM, and where gaps are identified to exist, take inspiration from literature to meet these absences, which will be identified through detailed analysis techniques, consequently helping to eliminate the acknowledged shortcomings of the existing methodologies to produce an end product which is more resilient to the common software development project failures.

To ensure that the project aim is comprehensively focused on producing results which are accurate and supportable, a number of objectives must be met throughout the project.

Objectives that will be completed through secondary research are:

Objective 1

- Identify the primary lifecycle and management reasons as to why software development projects fail.

Objective Purpose:
The primary objective aims to discover the leading issues that face software development projects to have them be considered a failure. Equipped with knowledge and understanding of what causes a project to fail it will enable the work carried out to address these specific problem areas, reducing the opportunity of failure to exist in future projects.

Objective 2

- Carry out a literature review which provides an overview of XP and Scrum and which critically evaluates how each practice within the methodology could support the identified areas of failure.

Objective Purpose:
This objective aims to provide an in depth understanding of the two aforementioned methodologies highlighting the characteristics of each and critically analyse how they overcome the identified problems or where they fail to provide support. With an enhanced knowledge and understanding it will permit the avoidance of identified issues with each methodology thus allowing the construction of a new, more structurally solid framework to deal with risks. Learning additional information about these methodologies will help formulate ideas as to how to go about the combing process.

Objectives that will be completed through primary research are:

Objective 3

- Carry out a gap based analysis of the selected methodologies.

Objective Purpose:
Within this objective the elements which make up the practices implemented by both methodologies will be analysed against how effectively they support the identified common failure areas. The results of this analysis method will provide further data as to which problematic areas are the least supported thus helping to create a prioritised list of failure areas.

Objective 4

- Carry out analysis of selected methodologies using a four dimensional analysis tool.

Objective Purpose:
The aim of this objective is to analyse each methodology from four different aspects; Method scope characterisation; Agility characterisation; Agile values characterisation and Software process characterisation. The data gathered from each of these four dimensions will provide textual data which can be simply compared to illustrate the capabilities of both methodologies and thus helping to recognise in which areas are more effectively supported by which method.

Objective 5

- Produce a questionnaire that can provide support to the research carried out.

Objective Purpose:
The questionnaire which will be created to satisfy this objective will be constructed based on the findings of the literature review completed in objective 2. The research completed in the literature review will provide information as to the areas which the questionnaire must inquire about. The questionnaire will be completed by a sample population who are involved with software development methodologies within their professional career.

Objective 6

- Amend practice elements to aid in the development of a revised process approach.

Objective Purpose:
The completion of this objective will require a revised process approach to be created based on the results of the data collected through completion of prior objectives. This revised methodology will aim to be more supportive of the identified failure areas by combining the strengths identified of either methodology thus making it a more effective approach from a development and management point of view.

Objective 7

- Evaluate the revised methodology by performing identical analysis as in objectives 3 and 4 to illustrate improvement.

Objective Purpose:
To complete this objective the revised approach will be analysed using the same methods as the original two methodologies were analysed. The completion of this analysis will enable the opportunity to illustrate that the revised approach is more supportive in overcoming the identified failures that were previously poorly sustained.

1.3 Report Structure

The remainder of the interim report will follow the format explained below.

Literature Review
The literature review completed in section 2 will contribute information as to the definition of failure and what are considered to be the most influential causes for software development projects to be considered a failure. The literature review will go on to give an underlying knowledge of each of the practices that are implemented within the methodologies of Extreme Programming and Scrum. Each practice will be critically analysed in relation to how it can help to provide a solution to overcoming the previously identified failures.

Methods
The methods section of this document will outline the methods which will be implemented to carry out the further research of comparing two agile methodologies and how they can be used to find a solution to problematic areas of software development projects. This section will also detail, in depth, the specific analysis tools which will be used to conduct this further research.

Presentation of Results
Section 4 of this report details the results gained from the research that was carried out within this project. The section will exhibit both quantitative and qualitative data gathered from research before discussing and evaluating how the results which are presented provide

benefits or dis-benefits in relation to overcoming the recognised project failure areas which this projects aims to address.

Conclusions and Discussion

The final section of this report will provide a summary of this research projects primary aim before detailing the conclusions which have made based on the results gained and illustrated in Section 4. The conclusions which are derived will then be analysed as to what they mean and how they relate to the work completed by others in similar research areas and also how the results produced relate to the projects research question. Lastly, this section will detail the limitations which were found to exist within the study, what could have been completed differently to further enhance the research and what potential areas of further research could arise as a result before drawing on a final conclusion.

2.0 Literature Review

The aim of the literature review is to research and analyse the arguments of others, to complete the objectives one and two identified in section 1.2.3 and address the questions which they pose:

- Identify the primary lifecycle and management reasons as to why software development projects fail.
- Carry out a literature review which provides an overview of XP and Scrum and which critically evaluates how each practice within the methodology could support the identified areas of failure.

The completion of these objectives will support the necessity for further research to be carried out with the goal of producing a unified methodology which will further reduce the possibility of common failures being influential in the failure of software development projects.

2.1 Failure

Whilst the technology implemented in software development projects continues to evolve at a high speed, generating a wider spectrum of opportunities for software developers to produce more complex, higher quality software, this achievement unfortunately cannot be attributed to the rate of success witnessed in IT projects, which continues to be a source of poor statistics.

2.1.1 What is failure?

Glass (Glass, 2005) explains that differentiating between success and failure can be difficult. He puts forth the point that a project may be "functionally brilliant" but it may not meet its budget constraints or time targets by a small percentage, can this be considered a success or should it be documented as a failure? The ambiguity makes it difficult to always distinguish between the two, and is a possible reason as to why project review reports can differ.

However, Sauer (Sauer, 1993) defines failure as "development of operation eases, leaving supporters dissatisfied with the extent to which the system has served their interests". This definition is more specific than Glass's but again is open to scrutiny as different companies will have dissimilar tolerance levels.

While finding a definitive definition of what constitutes as failure in an IT project is difficult, a single point which can be agreed upon is that, measures must be taken to ensure that the opportunity for potentially problematic factors to exist is reduced as much as possible.

2.1.2 Present problems in Software Development?

The cancellation or failure of a project is an unwanted incidence which can often result in a company's loss of economic stature and available resources, the embarrassment caused to those involved, and the potential result of an organisations extermination altogether (Ahonen& Savolainen, 2010).

To highlight that there exists a problem in IT projects, a study by the University of Oxford found that of the 421 projects studied, 35% were identified to finish behind schedule; 59% were found to finish over budget; and 54% of projects under delivered on initial commitments (Cuthbertson& Sauer, 2003). The identified difficulty of developing and maintaining software has become a highly problematic area in software engineering and the dismal statistics which accompany the problem has led to the current situation being called a "crisis" (Krishnan et al., 1999). This poor situation that software development projects find themselves is because the modern business must constantly work to stay ahead of its competitors and as a result the environment of software development is constantly changing (Hughes& Cotterell, 2006).

Much research has been carried out to identify the areas of software development which play a leading role in the projects which are regarded as a failure. Many studies have been carried out, each concluding several factors which are deemed responsible in the failings in software development projects. Three studies which have been completed (Cerpa& Verner, 2009); (Cuthbertson& Sauer, 2003); (Hughes& Cotterell, 2006) share many identified failings, illustrated in Table 1.

Table 1 Identified Potential Failure Causes

FAILURES	RESEARCH 1	RESEARCH 2	RESEARCH 3	TOTAL
Behind schedule	•	•	•	3
Over budget	•	•	•	3
Lack of knowledge regarding development area		•		1
Risks not continually re-assessed or controlled	•	•	•	3
Lack of quality standards		•		1
Lack of up-to-date documentation	•	•		2
Preceding activities not completed		•		1
Lack of techniques for measuring progress	•	•	•	3
Lack of communication	•	•		2
Poor role definition		•		1
Poor leadership	•			1
Incorrect success criteria			•	1
Project affected by deadline pressure	•	•	•	3
Staff unrewarded for working long hours	•			1
Client not involved in making project schedule	•	•	•	3
Poor change control	•			1
Unrealistic expectations	•	•	•	3
Insufficient review process	•	•	•	3
Changing project scope	•	•	•	3
Client not making sufficient time for requirement gathering	•	•	•	3

The information illustrated by Table 1 concludes that there are ten software development project failings that are identified by each study and therefore represent the problem areas which will be aimed at overcoming within this research project (Table 2).

Table 2 Top 10 Software Development Project Failures

FAILURE ID	FAILURE
PF1	The project was underestimated.
PF2	Highlighted risks were not continually re-assessed, controlled or managed effectively throughout the project.
PF3	The delivery date was often too ambitious, negatively impacting the delivery process.
PF4	Scope changed regularly throughout the project.
PF5	Review process at the end of each phase was not extensive enough or did not exist.
PF6	Clients not involved in the making of schedule estimates.
PF7	Clients did not make adequate time available for requirements gathering to take place.
PF8	Projects were often behind schedule.
PF9	Projects were identified as being over budget.
PF10	The development team under delivered on initial commitments.
	NOTE: 'PF' equates to Project Failure.

2.2 Extreme Programming

2.2.1 Overview

Created by Kent Beck, Extreme Programming (XP) bases itself on four fundamental values which must be upheld if success is to be achieved; Communication, Simplicity, Feedback and Courage. The concept of XP is based on that of an iterative methodology where the project is broken down into small phases, allowing the project team to work on "a little at a time" (Beck, 1999) throughout the entire software development. Ambler (Ambler, 2002) clarifies that XP is aimed at being used by small sized teams which operate in a hostile environment, that is to say one in which the requirements of the project can change at any time without prior warning. The author goes onto state that XP provides the promise that it will focus on the areas of the project which matter most that day.

2.2.2 Practices of XP

Each practice of XP can offer benefits to a software development project if implemented correctly. In this section of the literature review, an overview of what each practice consists of will be given and an analysis of the positive consequences each can have on a project, reducing the probability for those problems identified, to exist.

1. **Sit together**

Objective:

The purpose of this practice is to create a unified team ethic within the project group by ensuring that increased face time is made available. Beck (Beck, 2004) states that by proposing the idea that a team sit together to work, this will encourage conversation which will help team members to create amalgamated ideas and overcome obstacles faster (Hussain et al., 2008). However, he goes on to state that adopting this practice must be met with care as people can often feel vulnerable if their privacy is taken away abruptly, therefore the transition must be gradual.

Relation to Finding Solution:

Braithwaite & Joyce (Braithwaite& Joyce, 2005) explain the value of quality communication and state with it the project will become more productive in its processes. In addition, Millett (Millett et al., 2011) states body language is important in effective communication and that people will be more comfortable and trusting towards someone they communicate with regularly face to face rather than someone whom they communicate to only through voice, such as telephone, thus helping to make team members interaction more comfortable.

The concept of Sit Together also brings with it the beneficial experience of "osmotic communication" (Cockburn& Williams, 2000). Osmotic communication is the picking up of background sounds without being consciously aware, as though by the process of osmosis. When something recognised is voiced or a point that may be recalled from an earlier instance this causes a person to subconsciously become attentive and able to join in the discussion and provide an input. The benefit that this provides is immediate and rich

feedback thereupon lowering the cost of project communication and enhancing the learning of individuals within the team.

Additionally, Whitworth (Whitworth, 2008) supports this increase in effective communication by explaining that the continual engagement among team members will inspire them to feel closer to the project group strengthening their positivity and work ethic.

Whilst this practice does not bring about a direct solution to the problems identified in 2.1.2, this practice does contribute many advantages such as those discussed thus enabling it to help strengthen the development process.

2. Whole team

Objective:

"Include on the team people with all the skills and perspectives necessary for the project to succeed" (Beck, 2004). The purpose of this practice is simply as the name suggests, in order completing the project successfully, fulfilling the demand for high quality, it is necessary that for each required task in the project there is someone available within the team that has the ability and skillset to complete it. For a team to be considered "whole" Beck (Beck, 2004) explains that there has to exist diversification and that should a skillset no longer be required in the project then the responsible person is no longer required to be part of the team, for the primary function they were brought on-board for, and is therefore free to leave or continue working on an alternative task.

Relation to Finding Solution:

A key fundamental value to the success of implementing the practices of XP is communication. Juric (Juric, 2000) elucidates that XP promotes the use of structured communication among the entire team. As the team is made 'whole' by the participation of the team as well as the client this provides the opportunity for constant communication to exist hence why XP, as an agile process, deals well with adapting to the varying environment.

Moreover, this practice enables the developers to have a closer relationship with the client, thus allowing the team to develop and build a product which is to the requirements specified by the client rather than an assumption made by the development team based on gathered requirements. This therefore helps to accommodate a resolution which can overcome Problem 6 and Problem 7.

3. Informative workspace

Objective:

Shore &Warden (Shore& Warden, 2007) state that an informative workspace should transmit the essential information relating to the project into the room, and that it should provide the team members with information relating to how the project is progressing, highlighting the status of project tasks.

Beck (Beck, 2004) explains that the most effective method for enforcing this practice is to situate visuals around the development area which offer team members the necessary information they require. Shore &Warden (Shore& Warden, 2007) reinforce this method of active information but add that the importance of "hand-drawn" visuals should not be underestimated; as it is believed that it provides a more intimate way of communication which is more likely to have a stimulating effect upon people within the team.

Relation to Finding Solution:

The advantages brought about by implementing this practice can be of great wealth to the success of a project. An analytic study carried out by the Open University focused on an XP team using this practice and found that the information detailed on the several visuals located around the workplace was both easily accessible and "immediately relevant and applicable" to the team members (Sharp et al., 2006), promoting deeper understanding of the project and awareness of project progress.

Having a communal area of the workplace specifically designated to transmit key information relating to the project as well as providing information of the project progress allows for everyone to fully understand what has been done and what still is to be worked on, providing a means of communication that is visual rather than oral or documented. Being able to clearly see what has still to be developed within the sprint can allow for the development team to adjust so as not to fall behind the agreed upon work schedule, helping to resolve Problem 8. The ability to communally share information also provides a way in which to communicate the changing scope within the project, supporting Problem 4.

This space can be used not only to highlight the progress of the project but can be used as a means of communicating to project team the risks that have been identified throughout the project and what is being done to mitigate them. This can be updated by team members should the status of a risk change. It is this benefit which provides a method of overcoming the problem of tracking and controlling risks (Problem 2), an area which was found to be involved in the failings of many projects.

4. Energised work

Objective:

Beck (Beck, 2004) explains that in order to fulfil this practice a person should only work for as long as they are productive and states that by trying to show "commitment" by working overly long hours brings about more disadvantages to the project than advantages. Shore & Warden (Shore& Warden, 2007) add that by trying to complete work when not fully energized will cause mistakes to be made which can result in the creation of more work as the errors will then have to be corrected ergo delaying development time and increasing costs. This idea of over working is supported by Yourdon (Yourdon, 1997) who describes the attempt to complete unrealistic goals by working disproportionate hours as a "death march".

Beck (Beck, 2004) discusses the idea of the 40-hour week, which should enhance the team members' productivity by demanding they work a shorter period of time therefore helping

to meet the requirements of this "human focused" (Glass, 2001) methodology. Re-calibration can also play an important role in ensuring that the practice of energized work is upheld by re-planning work commitments so as not to overload the team members with work, which could be demoralising (Martin et al., 2009).

Relation to Finding Solution:

The practice of energised work focuses on the human aspect of software development and exhibits the idea that whilst communication among a team is vital in ensuring that a product meets the client's expectations of high quality, it is equally important that the morale of team members and their interest and motivation to produce a high quality product is remains high (Hunt, 2006). This practice consequently can work towards generating a resolution to Problem 3, relieving the team's stress of an upcoming delivery date by decreasing the amount of errors which may be made by over worked team members thus causing a reduction in time spent amending the software before release.

5. Pair programming

Objective:

Pair programming is a defining practice of XP and its objective is that all code written is completed by two programmers sitting at one machine (Beck, 2004). He goes onto explain that it is a method between two people simultaneously producing code, one person writing the code whilst the other constantly peer reviews it.

However, the practice of pair programming does capture more discussion than any other XP practice and in the early life of XP people were cynical to the idea of having two people work on one machine and that it was seen as intrusive to a person's personal space (Beck, 2004). Conrad, (Conrad, 2000) describes it as being a "controversial aspect" as it is believed by many that it cannot be cost effective having two people work on a single machine.

Conversely, a study found that programmers spent 15% longer on the programming of code than a single person would have but by using the practice of pair programming there was a 15% decrease in the number of defects identified in the code (Cockburn& Williams, 2000). They also state that due to the lower number of defects the costs overall do not increase as little time is spent correcting errors. To further highlight that the practice of pair programming is advantageous Williams & Kessler (Williams& Kessler, 2000) found that 96% of people using pair programming enjoyed their work more.

Relation to Finding Solution:

The synchronisation between a pair can become very valuable towards the team and the quality of work produced because the process of considering several strategies to overcome an issue is made simplistic by having the skills and knowledge of two rather than one (Shore& Warden, 2007). The practice also increases the confidence of those involved in pair programming and the quality of work produced is seen to be significantly higher quality than work completed by an individual (McDowell et al., 2006).

The adoption of pair programming brings with it a greater uniformity in the work completed (Rasmussen, 2003), whereas work completed by separate individuals will not match the same standard of quality when the work is finally put together as each will have an altered style. Additionally, pair programming also allows the programmers to learn from each other in a much more efficient manner than if they were trying to learn a new skill themselves and it is this rich learning tool that has seen this XP practice gather a much greater following in recent years (Lindstrom& Jeffries, 2004).

Pair programming has been found to bring the benefit of enhanced communication, among the team, through a more intense working environment (Highsmith& Cockburn, 2001). Further research carried out supports this and explains that speed of development of having two people working towards a single piece of functionality is no slower than an individual working on the developing the same functionality, but the quality of what is produced when pair programming is undertaken can be significant.

This practice primarily tackles the problem of poor communication within the team, as team members are forced to work together. However, as Rakitin (Rakitin, 2001) points out, this is only going to happen if the individuals involved in pair programming are experienced in using this method of development or have good communicative skills. The second identified failure that this practice could address is helping to reduce the negative brought about by Problem 3, overly ambitious delivery dates. By not losing any time with development but gaining the benefit of greater product quality (Reifer, 2002) this reduces the time that the team must spend on correcting errors at the end of the development process thus relieving stress to have the product complete by the set deadline. In addition, this practice demanding constant peer review and whilst this does not link directly with Problem 5, insufficient review of the project, it does enable an short term review to exist strengthening the development process.

Whilst there are many possible benefits to the use of pair programming it does not come without potential problems. One such problem of this XP practice is the implementation of it in different cultures around the world. As Beck (Beck, 2004) describes, some cultures it is acceptable to closely together in terms of physical distance whereas other cultures they expect no one to come within their personal space and doing so can be seen as intrusive. Further to this, Glass (Glass, 2001) states that pair programming is not suitable to everyone and that some people if put in the pair situation will not flourish as intended.

6. Stories

Objective:

Stories are simplistic but effective tools used to help the team understand what exactly they should produce by providing small descriptions of everything the stakeholders wish the team to make (Paulk, 2001). They are not purely a list of requirements nor are they use cases but each small description must be “business-orientated, testable and estimable” states Beck (Beck, 1999).

As described earlier in the section relating to the informative workplace, the stories that are created should be put up around the workplace to allow for all team members to view and

learn which stories have had work produced for them and which still need to be completed (Beck, 2004). Glass (Glass, 2001) also puts forth the point that the stories should not be written in technical jargon but that they should use everyday language to portray simple ideas to the team.

Relation to Finding Solution:

The method of using stories to convey the aims and tasks of a project provide a simplistic connection between the client and the project developers (Ming Huo et al., 2004). The process of gathering requirements is one of the most important and complex tasks within a project. The client, who is unlikely to have a rich technical understanding, must convey their desires to the developer in a clear manner. By implementing this practice the client can portray their wants in a simplistic manner (Khalaf& Al-Jedaiah, 2008) thus aiding the process of defining project requirements, an area which that Problem 7 identified to be a source of failure among software development projects.

7. Weekly cycle

Objective:

An aim of using XP in the development of software is to produce a section of working software every week (Shore& Warden, 2007). The advantages brought about by encouraging this strict discipline among developers is of great worth to the outcome of the project. Beck (Beck, 2004) explains that in order to meet this ambition and make the development process as simple as possible for developers, it is beneficial to hold a meeting which discusses the clients beliefs relating to the order of priority of the stories created. When the stories which are to be developed over the forthcoming week have been selected they should then each be broken down into smaller tasks that can be completed with greater ease.

Highsmith & Cockburn (Highsmith& Cockburn, 2001) support this weekly preview by highlighting that this gives the client an opportunity to provide input and notify the team of any changes which they believe are needed.

Relation to Finding Solution:

As Problem 3 recognises the delivery date in software development projects is often too ambitious thus having a negative impact on the product delivery process. By producing a piece of working software every week it provides a means of measuring project progress against predetermined timescales. If it is highlighted that the project is behind in terms of time, then measures can be taken to help ease the strain, such as asking the client for an extension of time or for the allowance in the reduction of functionality that must be present by each release, providing support to this problematic area.

The Weekly Cycle can also make the process of an adjusting scope much easier to deal with from the project teams' point of view as the meeting can provide an opportunity for both parties to discuss any alterations to the scope therefore helping to avoid Problem 4.

Lastly, the Weekly Cycle can make available a set time to not only preview what is forthcoming but can allow for a review point of what has just been completed. Whilst the Quarterly Cycle has a more in depth review process, a weekly review can produce short term information relating to progress and the controlling of project risks, which will be helpful as Problem 5 identifies that there is often too little opportunity for project review to take place.

8. Quarterly cycle

Objective:

The quarterly cycle adopts a similar concept as that of the weekly cycle but instead looks to evaluate the previous three months in terms of project progress and look forward to the forthcoming three months contemplating the larger goals which must be met (Beck, 2004). Maurer & Martel (Maurer& Martel, 2002) explain that this quarterly meeting allows for the client to regularly see first-hand what has been produced and provide feedback, a founding value of XP, thus reducing the risk of incorrect functionality being developed.

Relation to Finding Solution:

The Quarterly Cycle accomplishes the objective of being an effective tool used to enable the tracking of project progress on a long term basis. By gaining a wider overview of data relating to project progress, this can allow the team to re-evaluate how their time is managed, thus providing a key step in overcoming the acknowledged failure of a project falling behind schedule (Problem 8).

Further to aiding the re-evaluation of resource management, the Quarterly Cycle also contributes a chance to review risk management and the product, as a larger entity. Feedback is an important commodity which helps to identify problems early on, avoiding the painful experience of creating a product which does not meet the expectations of the client (Hunt, 2006). This ability to review such areas with the additional aid of a great depth of data allows the project team to constructively find solutions to combating recognised failures such a poor phase review and supervision of existing and potential risks, providing a method in which to overcome Problem 2.

9. Slack

Objective:

The practice of slack provides the idea of a buffer zone for the development process. It is wasteful to overcommit to work and then under deliver (Beck, 2004), because it will mean the distribution and effectiveness of resources is not being maximised. In order to allow a development to be as effective as possible the team members need time to be able to think and create solutions. DeMarco (DeMarco, 2002) asserts that the primary resource required for innovated development is slack; if a team cannot create a solution then it is likely that they are being made to work too hard on other areas of the development and not being given a chance to brainstorm a solution.

Relation to Finding Solution:

Problem 1 identified that underestimating a project can have serious consequences not only in the development process but also to the organisations involved; often resulting in the project overrunning time constraints thus resulting in the requirement to spend more money to ensure that the project is completed, supported by the figures provided by Cuthbertson (Cuthbertson& Sauer, 2003). The practice of slack provides the project team with a buffer should they underestimate a project. Having additional time which can be used means that the project can overrun a little without having serious consequences as well as releasing pressure on team members resulting in their quality of work being of a much higher quality.

10. Ten-minute build

Objective:

The ten-minute build practice of XP states that the automated building of the system and the running of system-level function tests should not take more than ten minutes to complete (Beck, 2004). The build process should be comprehensive but not overly complicated to gain the greatest benefits. If the build process is recognised as being slow then it is likely that the tests in place are the reason and should therefore be replaced by more maintainable unit testing methods (Shore& Warden, 2007).

Relation to Finding Solution:

The benefit of the Ten-Minute Build is that the developer is able to build and test the entire project almost instantly. This rapid production can allow the developer to receive rapid feedback detailing that the product works and is without errors thus reducing the time spent having to debug or correct errors at a later stage in the development process consequently providing the potential of increasing production time, helping to ensure that a solution to Problem 3, having overwhelming delivery dates, can be created. Furthermore, this practice also provides the developer with a way in which to gain rapid feedback as to the current software enabling them to review, albeit in a short-term basis, what has been constructed.

11. Continuous integration

Objective:

Beck (Beck, 2004) explains that when developing software it should be integrated and tested regularly with existing code, every couple of hours. By integrating and testing continuously it allows the risk of an “integration hell” (Lindstrom& Jeffries, 2004) to be avoided as any errors in the code will be detected before they can escalate into a more serious situation. Rasmusson (Rasmussen, 2003) supports this by saying that the continual integration and examination of new code allows for the traditional implications that are associated with integration to be avoided.

Relation to Finding Solution:

Continuous integration reduces the chance for risk to exist within the produced product through constant error checking each time a new piece of code is added to the already

existing code. This method of integration can help to minimize the need for developer to go back to the code a later stage in the project when a deadline is close thus helping to make the task of delivering on time much easier (Beck, 2004).

Maurer & Martel (Maurer& Martel, 2002) put forth that an additional benefit of continuous integration is there is always an available executable version of the software which can serve as a baseline for all future work. Paulk (Paulk, 2001) continues to support this continuous integration by stating that it prevents the software functionality retrogressing when the requirements are altered.

An additional benefit of continuous integration is that by correcting errors as soon as new code is integrated reduces the potential risk that could evolve and therefore does not have to be constantly monitored, an area that has been identified as being large element in software development failures.

12. Test first programming

Objective:

Test first programming is the practice of creating a test based on the requirements that the software is expected to achieve prior to the creation and implementation of any code (Rasmussen, 2003). The objective of this exercise is to provide a focus when coding the next step of the software and ensuring that each piece written works in conjunction with previously written code (Beck, 2004).

Research has shown that implementing this method in the development process increases productivity (Erdogmus et al., 2005) and that the quality of code written is enhanced thus resulting in a reduction for the need for alterations later on in the development process.

Relation to Finding Solution:

This practice would be highly beneficial in finding a solution to the identified problem of risks not being appropriately controlled and managed throughout the project, recognised as Problem 2. As the purpose of this practice is to create a method for evaluating the product within the iteration before the product is constructed strengthens the validation process. Allowing for time to carry out this practice will help to reduce errors within the product (Siebra et al., 2008) thus helping to reduce the opportunity for risk to exist.

2.2.3 Benefits Summary

XP provides many potential solutions to the problems identified in 2.1.2 but the methodology also contributes largely to producing more effective communication to exist amongst the development team. Although, poor communication is not within the identified ten listed failures it is still a constant threat to all team based projects and therefore the support shown towards communication from XP can only help to strengthen the development process.

From the literature reviewed relating to the practices of XP, the benefits that each practice can contribute to overcoming the identified problems, which have been associated with the failure of many software development projects, have been summarised in Table 4 on the following page.

Table 3 Summary of solutions brought by XP Practices

	Sit Together	Whole Team	Informative Workplace	Energised Work	Pair Programming	Stories	Weekly Cycle	Quarterly Cycle	Slack	Ten-Minute Build	Continuous Integration	Test First Programming
Problem 1									•			
Problem 2	•		•		•			•		•	•	•
Problem 3				•	•		•			•		
Problem 4	•		•				•					
Problem 5					•		•	•		•		
Problem 6		•						•				
Problem 7		•				•						
Problem 8			•	•				•				
Problem 9		•						•				
Problem 10		•										

KEY:
• Identified as being able to support problem.
• Identified as being potentially able to support problem.

2.3 Scrum

This section of the literature review will identify and provide an explanation of the characteristics of Scrum. It will go on to discuss how each practice could potentially provide a solution in overcoming the problems that were identified in section 2.1.2, to be in some way responsible for the failure of several software development projects.

2.3.1 Overview of Scrum

Schwaber (Schwaber, 1995) states that the "closer the development team operates to the edges of chaos, while still maintaining order, the more competitive and useful the resulting system will be". It is this principle that has made Scrum appeal to software development teams and which has seen it become the most employed agile methodology. According to recent research which found that 58% of software development teams, adopting an agile approach, used Scrum (VersionOne, 2010).

Takeuchi & Nonaka (Takeuchi& Nonaka, 1986) realised that the method in which software products were developed was changing rapidly due to the introduction of new technologies available. This meant that the existing linear approaches were no longer suitable and that a tactic which mimicked the unsettled, turbulent environment was required. To express this need for flexibility and agility in development they coined the sport metaphor of rugby, where a team move forward together, as a single entity adapting to the environment with a single goal in mind.

It was this conceptual metaphor that Ken Schwaber and Jeff Sutherland developed further in 1995, naming the approach, Scrum. This methodology has key characteristics which stay true to the original metaphor and allow it to operate in unpredictable surroundings with intensity and speed (Schwaber & Beedle, (Schwaber& Beedle, 2002).

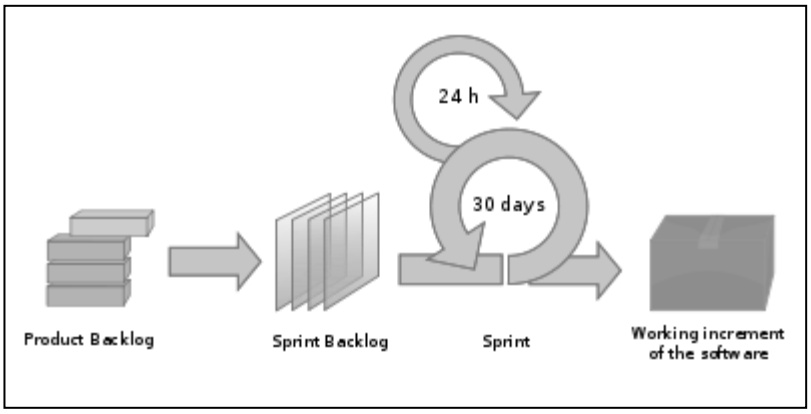


Figure 1 The Scrum project management method. Part of the image is based on public domain graphics from Open Clip Art Library - Lakeworks

As is in the sport of rugby, a scrum consists of a team of eight individuals all working together for concentrated periods of time in order to achieve a unified goal, the production of working software after thirty days of development (Rising& Janoff, 2000), exemplified by Figure 1. Schwaber & Beedle (Schwaber& Beedle, 2002) describe Scrum as working at a "team level" thus requiring that, again like rugby, each member must fully understand their

own role within the team and be clear as to what they are trying to achieve resulting in the ability to create complex, higher quality products in a shorter period of time.

2.3.2 Scrum Practices

To gain the benefits that Scrum can offer, it elicits a set of practices that, if adopted, allow for the efficient development of a quality product in an uncertain environment. An overview of these practices follows accompanied with analysis as to how they can relate to providing a solution to the problematic areas of software development identified in the previous section.

1. **SCRUM Master**

Objective:

The definition of the Scrum Master's role, as given by Schwaber & Beedle (Schwaber& Beedle, 2002) is; "being responsible for the success of scrum". The Scrum Master exists to ensure that that values, principles and practices that exist within Scrum are implemented throughout the entirety of the development process. The Scrum Master's role is not to play that of a traditional manager due to the belief of the founders, that the traditional manager has a predisposition to work via authority rather than influence which would prevent the Scrum team from achieving a key characteristic of self-management (Yi, 2011).

The secondary role of the Scrum Master is to certify that everyone related to the development process, has the opportunity to voice opinions and provide solutions to overcoming obstacles which may exist within the development process. Recording the decisions made and keeping track of development progress is also an important task which must be fulfilled, thus helping to avoid the risk of bad tracking.

Relation to Finding Solution:

As described, the primary role of the Scrum Master is to enforce that the processes and principles of Scrum are upheld. Problem 2 and Problem 5, as identified earlier, can be better supported as (Rising& Janoff, 2000) stated the Scrum Master is able to donate a greater amount of time to the monitoring of the tasks responsible for providing support, thus increasing the likelihood that the risks identified can be constantly tracked consequently reducing the potential adverse consequences which may develop within the project.

Additionally, the adoption of the scrum master role will also help to increase the effectiveness of communication among the team through the strict control of daily scrum meetings. Incorporating discipline at these meetings, the scrum master can encourage effective communication between team members allowing each to present their opinions and feedback relating to the project and to build positive relations and a method of gaining essential information.

Lastly, Hong (Nayoung Hong et al., 2010) explains that by having a Scrum Master it helps the team to become acquainted with the other practices of Scrum, helping to understand their

benefits and aiding the team to be able to execute them more effectively which will support the opportunity for an increased active development process.

2. Product Backlog

Objective:

The Product Backlog is a constantly changing, prioritised list of both, the business and the technical functionality that is still to be developed to complete the fabrication of an end product (Ota, 2010). Furthermore, the Product Backlog contains every component that those connected with the development feel is required in order for the product to be deemed complete (Schwaber& Beedle, 2002).

As the Product Backlog is such a central element to the success of Scrum it is essential that it is maintained and updated in parallel to the products development, this responsibility lies with the Product Owner (Tuomikoski& Tervonen, 2009).

Moreover, it is vital that the 'stories' used to describe each component of the Product Backlog is properly formed and has requirement in order to be suitable for consideration of development within a sprint. Poorly formed elements of the Product Backlog will be rejected until they are improved thus eliminating the chance of confusion during the sprint period (Sutherland et al., 2009).

Relation to Finding Solution:

Adopting the practice of the Product Backlog can potentially address three issues identified to be linked to failure in software development projects. The first problem which this practice can provide a solution to overcoming is the underestimation of a project, Problem 1. Containing all the required components which are considered to be required in the project, the project team and the clients will have a greater opportunity of accurately identifying the depth of the project and therefore what resources are needed for it to be completed.

The second problem area that the Product Backlog can help to overcome is Problem 4. It is inevitable in modern day software development projects that the scope will often change to help the client to maintain their "competitive advantage" (Beck et al., 2001). Therefore a document which keeps updated records of all the clients requirements and which is available to each team member is of great value and ensures that each team member understands what is being asked of them. Additionally, the document will provide a prioritised list thus ensuring that the components considered to be the most important are completed first.

The Product Backlog offers a third positive attribute, helping the scrum team to overcome the identified weakness of Problem 7. The importance of strong requirements gathering is vital to the success of any project and as Najafi (Najafi& Toyoshiba, 2008) explains, gaining knowledge of what the client wants from the end product helps to prioritise each feature in the Product Backlog. By representing each requirement with a small, simple story, it

furthers the effectiveness of the Product Backlog as each team member can find it more understandable and relatable rather than a document containing technical jargon.

3. SCRUM teams

Objective:

The Scrum Team commits itself to the achievement of completing the chosen set of components, from the Product Backlog, selected during the Sprint Plan Meeting, thus producing a sprint goal (Schwaber& Beedle, 2002). Whilst the Scrum Master holds the overall responsibility of the Scrum, the Scrum Team has the liberty and authority to self-organise, completing the tasks of the sprint in a manner they deem suitable. Turk (Turk et al., 2005) explains that by being self-organised, a team will select a group of professionals that offer diverse skills therefore giving greater opportunity to build greater quality products. This is supported by Rong (Rong et al., 2010), clarifying that having a team built from motivated and skilful participants are more likely able to successfully uphold the process evolution.

A further aspect of the Scrum methodology which is a key element in its success is the size of the team. As is in rugby, a Scrum consists of eight players and this idea has been carried over to Scrum. Supporting this, Miller (Miller, 1994) suggests that to optimise the performance interaction and communication of a team whilst still meeting the productivity demands is seven, plus or minus two.

Relation to Finding Solution:

The intention of achieving superior communication brought about by using Scrum will help both the project team and client to have a truthful estimation of development time rather than an assumption based on the client's expectations thus supporting Problem 3. By collaboratively coming to an agreement of a timescale the Scrum team will have a positive motivation as to the project deadline subsequently making them more confident in committing to the delivery of several components within a Sprint.

Sutherland (Sutherland et al., 2009) found that having small, intimate teams helps the team members to gain a stronger image of themselves as a team rather than individual pieces of a production machine. He goes onto explain that greater team building was witnessed as a result of team members looking at each other's 'to do' list, giving them a better understanding of the project.

4. Daily SCRUM Meeting

Objective:

The daily Scrum meeting is a daily fifteen minute status meeting to allow for an overview of development status and to communicate any complications which may have arisen within development (Boehm et al., 2009). The concept of this meeting is for each team member to answer three questions, giving short summarised descriptions. These questions are, as stated by Sutherland (Sutherland, 2005):

- What tasks have been completed in the previous 24 hours?

- What obstacles were found when completing these tasks?
- What tasks are expected to be completed in the next 24 hours?

During the Daily Scrum Meeting it is the responsibility of the Scrum Master to enforce that the meeting keeps to its strict time limit by ensuring that everyone present speaks only briefly (Schwaber& Beedle, 2002). Any further discussions that an individual wishes to bring up relating to the project should organise a separate meeting, with only those whom the matter concerns thus allowing those not involved to continue with their work resulting in maximum resource efficiency.

Relation to Finding Solution:

The daily scrum meeting is an active solution to overcoming Problem 2. It can enable each team member to better understand project progress and receive information relating to emerging risks, accommodating risk and development tracking. Issues raised within the meeting could be solved quickly, as a team, often with a variety of solution paths (Law& Charron, 2005).

The meeting also presents a chance for the development team to be updated about any changes relating to the project scope which have been requested, thus helping to support Problem 4. This reduces the probability of confusion among the team, helping to avoid wasting time on unnecessary tasks.

5. Sprint Plan Meeting

Objective:

The objective of the Sprint Plan Meeting is to determine which tasks the Scrum team will work to complete over the following sprint period. Whilst traditional practices such as planning and documentation are not heavily used in Scrum it remains vital that everyone understands what targets should be aimed for. The Sprint Plan Meeting is divided into two separate meetings.

The first meeting includes the Scrum team, management, product owner and the end users and the aim is to discuss and identify from the Product Backlog which features should be developed in the sprint.

The second meeting is attended by only those within the Scrum team. This meeting provides the team with the opportunity to determine how they will approach the sprint and how they will go about creating a solution which will enable them to produce the components which were selected from the Product Backlog. The meeting will also serve as a chance for the team to self-organise itself and delegate tasks to individuals for the sprint.

Relation to Finding Solution:

The sprint plan meeting will help the scrum team to overcome Problem 1 as the meeting will consist of a detailed discussion about which elements of the development are most important to the stakeholders therefore providing the development team with a prioritised set of requirements.

The meeting gives an opportunity for all team members to participate and ask any questions to grasp a stronger understanding of the sprints tasks as well as helping to commit towards a common goal (Paasivaara et al., 2008). From here the scrum team can set realistic and achievable goals which they aim to meet within the thirty day sprint period but which also keep in mind the final delivery date of the project which will consequently enable them to avoid under delivery on commitments, Problem 10, an area for which many software development projects have failed.

6. Sprint

Objective:

A sprint is a thirty day period which by the end, the Scrum team will have produced a tangible and usable product which contains the functionality that was decided upon at the Sprint Plan Meeting. The word sprint is defined as "a brief spell of great activity" (Dictionary, 2012) and it is this definition that it aims to emulate. Sutherland (Sutherland et al., 2007) describes a sprint as being a period of "hyper-productivity" in which the team aim to deliver their initial commitments to the customer.

To add to the intensity of a sprint, it is time boxed. This therefore means that the end date of the sprint, or delivery date, cannot be changed to accommodate delays from the Scrum team. Thus if a team feels that it has overcommitted itself for the sprint and that the meeting of their target seems unlikely they can reduce the implemented functionality but must still produce it by the end of sprint deadline.

Relation to Finding Solution:

The sprint provides a way in which the development team can avoid the failure of Problem 3 as having an excessively optimistic deadline date will inherently affect the development process putting the project team under much stress to complete the product and meet the quality expectations. Whilst a sprint is time boxed and as explained previously the deadline cannot be changed, the ability to reduce the functionality applied within the sprint period can allow the team to deliver a product which is of a much greater quality therefore reducing the negative impact brought to the project.

7. Sprint Review

Objective:

The Sprint Review Meeting occurs at the end of each sprint period. It provides an opportunity for the Scrum team to meet with all project stakeholders to showcase what has been produced within the sprint. This provides the stakeholders to give feedback on what they have seen and put forward any ideas that they may have, including functionality that they feel should be added or removed. The reprioritisation of elements may also be changed at this meeting should the stakeholders feel it is in their greatest interest to focus on a specific piece of functionality (Rising& Janoff, 2000).
The Scrum team can also deliver a summary of events within the sprint, describing the areas which were considered successful and the areas of the development process which posed

problems. As a whole this meeting is aimed at being constructive and evaluative helping to avoid costly risks which may harm the project.

Relation to Finding Solution:

The sprint review provides both the project team and the stakeholders of the project the opportunity to communicate and evaluate how effective the sprint was and what lessons can be learned for future sprints to avoid any problematic areas which may have occurred in previous sprints (Reel, 1999).

The review process can offer team members and project stakeholders the benefit of effective communication relating to the status of features and provided a chance to determine when there were gaps between features therefore allowing the team to re-evaluate the current situation and amend as necessary (Berczuk, 2007).

Through implementing this practice it provides a backing to overcoming Problem 5. The Sprint Review Meeting specifically accommodates this failure and is an active measure providing an opportunity for the stakeholders to communicate constructive feedback relating to the emerging product. Cerpa & Verner (Cerpa& Verner, 2009) relate the idea of an analytic review being similar to that of a "post mortem" whereby the aim is to distinguish where the faults were and why they occurred thus reducing the future project risks.

In addition to supporting Problem 5, the application of this practice will also help the team and stakeholders to revaluate the projects budget and scheduling lowering the opportunity for failure to exist thus aiding solutions to Problems 8 and 9.

2.3.3 Benefits Summary

From the literature reviewed relating to the practices of Scrum, the solution that each practice can contribute to combatting the identified problems, which have been associated with causing the failure of many software development projects, have been summarised in Table 3 on the following page.

Table 4 Summary of solutions brought by Scrum Practices

	Scrum Master	Product Backlog	Scrum Team	Daily Scrum Meeting	Sprint Plan Meeting	Sprint	Scrum Review Meeting
Problem 1		•			•		
Problem 2	•			•			
Problem 3			•			•	
Problem 4		•					
Problem 5	•			•			•
Problem 6	•						
Problem 7	•	•					
Problem 8	•					•	•
Problem 9							•
Problem 10		•			•		•

KEY:
• Identified as being able to support problem.
• Identified as being potentially able to support problem.

3.0 Methods

This section will describe and discuss the primary research methods which were selected in order to investigate the advantages and disadvantages that both Extreme Programming (XP) and Scrum provide towards offering greater support to the common project failures in software development projects, thus satisfying the primary research objectives documented in 1.2.3. The section will then discuss the technique which will be used to analyse the results produced from the analysis performed supplying an in-depth assessment giving greater academic rigour to the project outcome.

3.1 Primary Research Method

The aim of this project is to investigate and measure the prospective success of combining the strengths and ideas of two agile methodologies together to create a hybrid software development methodology which can increase the effectiveness of support given to the ten project failure areas without a decrease observed of the agile capabilities. To accomplish this goal the primary research method will be an evaluative experiment.

To gather data which will support the project hypothesis a quantitative research strategy will be adopted. The objective of this project is to collect factual evidence relating to the agility of two software development methodologies and compare the data to create a hybrid methodology which will test the project hypothesis. Quantitative research will be beneficial to the quality of the data collected thus strengthening the quality of the conclusion found (Naoum, 2006). Quantitative research is defined by Naoum (2006) as being an investigation into an identified problem area, based on the testing of a hypothesis measured by empirical data, using statistical processes to conclude if the project hypothesis could be true. The product of this research will be countable and measureable data which is required to provide evidence to support the hypothesis.

Oates (2006) describes the advantages of using an evidence based work strategy stating that by studying the work of other researchers, analysing and evaluating their research, a stronger foundation of knowledge is gained which can then be developed upon. Following the advice of Oates, a literature review has been completed to primarily provide an underlying knowledge of both methodologies and secondly to draw conclusions relating to the subject area based on the information put forth by other researchers. Figure 2 on the following page illustrates the tasks (depicted by a square) which will be completed in this project and the outputs that each task will produce (depicted by a circle) with each step being described fully with regards to its purpose and how it will contribute to the outcome of the project.

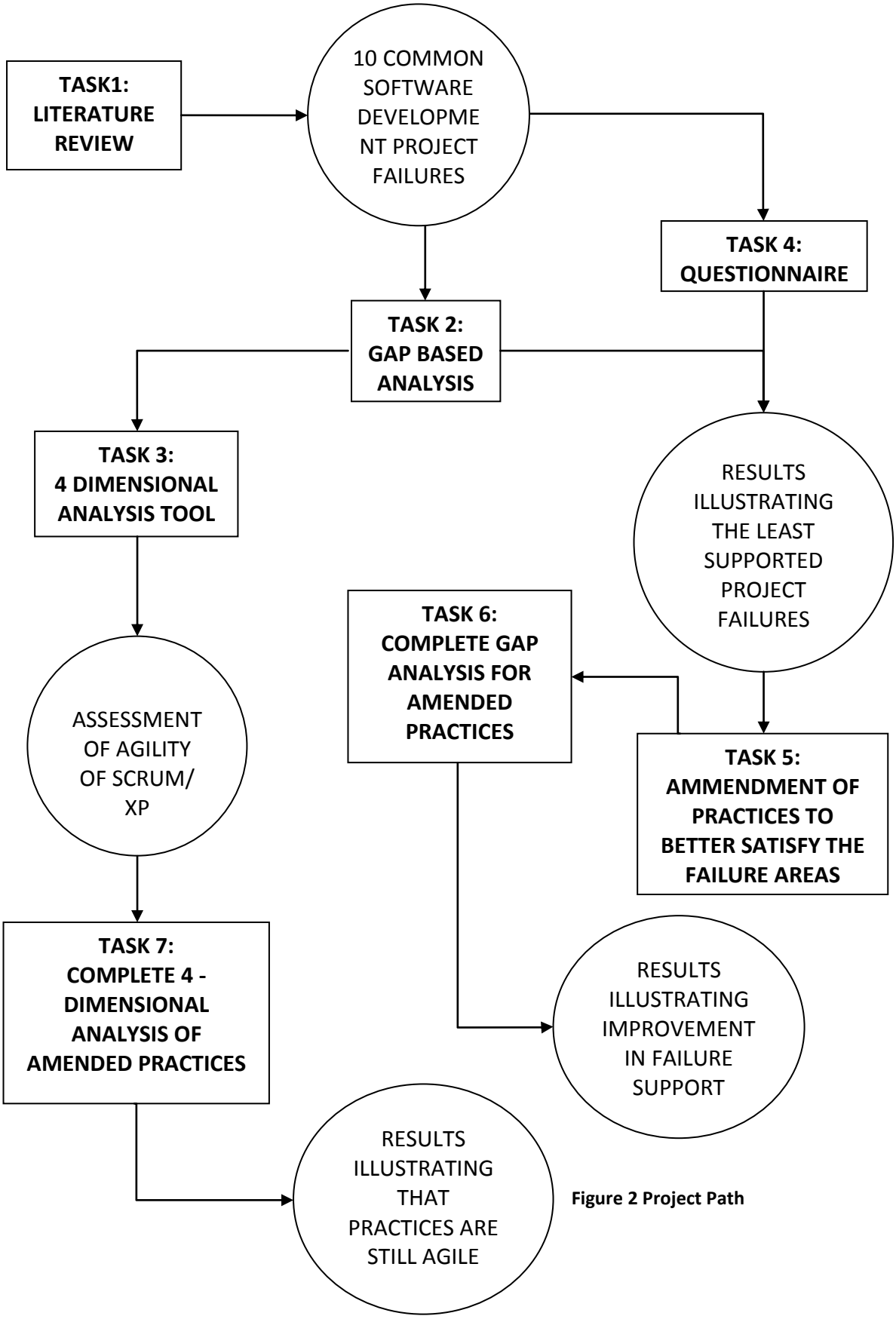


Figure 2 Project Path

The primary task upon completion of the literature review is to carry out a fine grained assessment of the how each practice within the two methodologies provides support towards the ten project failures. The purpose of a gap based analysis is to identify the discrepancies between the distributions and the integration of resources in both XP and in Scrum. The analysis will provide information relating to the either methodology's

capabilities and will determine how effectively each practice within either methodology supports the ten failure areas identified from the literature review.

Once an analysis has been completed for each methodology it will provide an illustration as to the failure areas which are the most poorly supported by either methodology. The results of this analysis will then be combined to produce an overall picture of the poorest supported areas and therefore will enable prioritisation of which areas must receive primary focus when revising each practice in attempt to make it more effective.

The second task which will be completed aims to provide greater academic rigour to this project with completion of a 4 Dimensional Analytic Tool (4-DAT). A study by Qumer & Sellers (Qumer& Henderson-Sellers, 2006), implemented a 4-DAT which possesses the key feature of being able to specifically provide a way in which to measure the agility of a particular software development methodology at a precise stage in a process, using any of the practices applied by a method. In the original study they analysed the agility of both XP and Scrum from four different perspectives, concluding that XP had a greater number of agile phases whilst Scrum had more agile practices within it.

Using the research by Qumer & Sellers (Qumer& Henderson-Sellers, 2006) as evidence based learning, it will act as a baseline against which the research in this project will be carried out. There were ten factors which have been identified in the literature review relating to the failings of software development methodologies which are not accounted for in the original 4-DAT study. It is for this reason that whilst the same concept of the 4-DAT will be adopted, several factors will be added or altered to provide an analysis which is specific to the current problems facing software development methodologies and which is specific to the investigating this projects research question.

In addition to the previous task, a qualitative method of research will be carried out in the form of a questionnaire. The aim of this is to gain primary sourced industry feedback from those involved in software development, particularly those who use XP and Scrum methodologies. The data gathered will provide further data which will reinforce the findings of the gap based analysis.

Upon completion of the three analysis methods, results that depict the failure areas which require the greatest attention, and results relating to the agile capabilities of both XP and Scrum will be generated. It is at this point that Task 5 shown in Figure 2 will be completed. Amendment of existing practices which are found to bring many benefits and capable of harnessing additional elements, which can further develop their effectiveness within their respective methodology, will allow for the methodology as whole to become stronger and more supportive towards the identified project failures. To identify if an improvement has been made, each method will be re-analysed using the same comparative analysis resulting in the completion of Task 6. Furthermore, to illustrate that either method is still agile in its nature, use of the 4-DAT will be implemented for a second time (Task 7), as a loss in agility would be detrimental to the outcome of the project as it would not meet the objective of allowing the methodologies to remain as agile methods.

3.2 Precise Nature of Research

3.2.1 Comparative Analysis

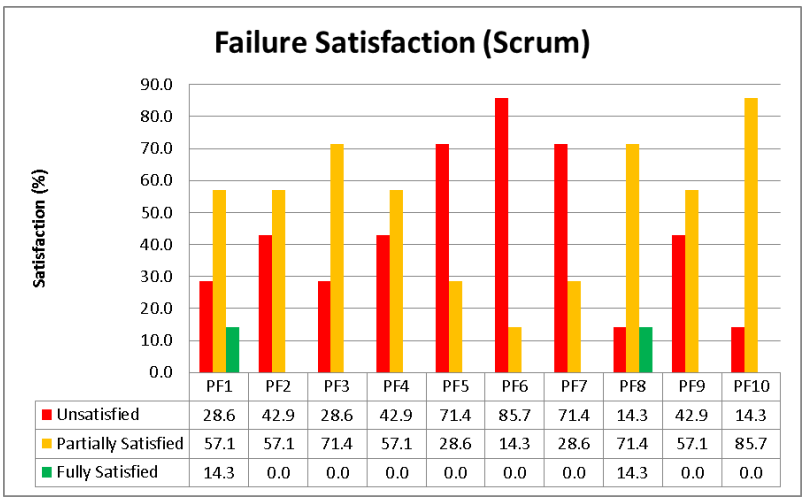
The primary method of analysis which will be utilised in the completion of this research project is a comparative analysis. The objective of this analysis is to measure how effectively each of the individual practices, contained within both methodologies, support the top ten software development project failures which were deduced from the work completed during the literature review. The concept which this method of analysis is based around, is a study completed by Abrahamsson (2003), in which several agile methodologies were compared, breaking them down into three categories and mapping them against the various stages involved within a software development.

Figure 3 illustrates an example of the analysis technique that will be used. As it demonstrates, this practice is broken down into three elements, each of which must exist for the practice to be effective as it was intended. If an element of the practice can provide a way in which to support one of the ten identified failures (Table 2) then it is marked with a dot. Once each element has been analysed against the failings, the failures are awarded one of three satisfaction levels inspired by Marcal (2007). Fully Satisfied (FS) is awarded if all elements contribute to the support of a failure, Partially Satisfied (PS) is awarded if only some of the elements of a practice contribute in some way to helping a failure and an Unsatisfied (U) is given if no individual elements within a practice can support any of the identified problem areas.

SCRUM REVIEW MEETING	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Stakeholders review developed product	•				•			•		•
Project team review performance	•	•	•							
Project re-evaluated	•							•	•	•
SATISFACTION CRITERIA	FS	PS	PS	U	PS	U	U	PS	PS	PS

Figure 3 Practices' element match against project failures example

The data acquired from this analysis can then be broken down to provide a set of results which can be illustrated in a simplistic graph to highlight the gaps which exist. The total number of points given to a failure is divided among the three categories (U, PS, and FS) with the total of each category calculated. The score awarded to each category within each failure area is then divided by the number of practices within the methodology being examined to provide a satisfaction percentage as shown in Figure 4, thus resulting in a graph being constructed, depicting clearly the failures that are strongly supported and those which are not.



	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Unsatisfied	28.6	42.9	28.6	42.9	71.4	85.7	71.4	14.3	42.9	14.3
Partially Satisfied	57.1	57.1	71.4	57.1	28.6	14.3	28.6	71.4	57.1	85.7
Fully Satisfied	14.3	0.0	0.0	0.0	0.0	0.0	0.0	14.3	0.0	0.0

Figure 4 Example satisfaction results of practices against failures

The analysis will produce two sets of results, one for each methodology, identifying how well they support the ten failure areas individually. These results can then be culminated to provide a third, overall analysis, identifying the problem areas most poorly supported by both methodologies thus providing the data to create a prioritised list which the re-evaluative tasks will aim to answer.

The results which will be gained from this analysis will clearly illustrate two important facts. The principal datum which the results will highlight is which of the individual practices, implemented by XP and Scrum; provide the greatest support to each failure and which contribute very little to overcoming the identified problem areas. These results will be valuable as they will help to determine the practices which may be best employed when constructing a hybrid process. The second outcome that will be derived from the data gathered is a prioritised list which depicts the failure areas with the most backing and those with the least, thus enabling the remaining work completed to focus more on the poorly supported areas which will help to reduce the overall factor of risk.

3.2.2 Four-Dimensional Analytical Tool (4-DAT)

From the literature review it was identified that there are several problems which exist in current day software development methodologies which have been found to be partly or wholly responsible for such projects to be deemed as failures. To be able to understand and evaluate the efficiency of current agile methodologies, effective analytic tools are required to be used.

The 4-DAT will be used as the secondary tool for collecting primary data in this research project, analysing the agility of both XP and Scrum methodologies, examining each methodology from four perspectives. These are:

1. Method scope characterisation
2. Agility characterisation
3. Agile values characterisation
4. Software process characterisation

<u>Dimension 1 - Method Scope Characterisation:</u>

The first dimension of the 4-DAT aims to identify the key characteristics of each methodology in relation to specific scope items which have been collated from each of the agile methods used, Extreme Programming (Beck, 2000) and Scrum (Schwaber & Beedle, 2002), giving a high level overview. The scope items used in this research project differ from the original study completed by Qumer & Sellers (Qumer & Henderson-Sellers, 2006) to reflect the initial conclusions and information gathered from the literature review thus enabling it to be more specific to this project. Table 5 illustrates the scope items which will be used and describes their purpose. As the literature review identified, XP contributes highly to the aspect of communication, therefore Scrum should be measured against this advantage to allow the possible identification of a 'gap' to be made. An additional scope item to the original study named 'Analysis Strategy' has been included in the first dimension as the findings of section 2.1.2 identified that the review process within software

development projects was poor and consequently further analysis is required to be carried out with regards to this area.

Table 5 Dimension 1 Values

Scope Item	Description
Project Size	What is the specified project size which it is believed that this methodology can support?
Team Size	What is the specified team size which it is believed that this methodology will be most effective at?
Team Structure	How is the project team structured?
Development Strategy	What is the development strategy (systematic, iterative, rapid) that this method implements?
Development Environment	How is the workspace specified to be most effective?
Communication Culture	What elements are in place to encourage team communication?
Analysis Strategy	When and how is project analysis and feedback used?

Dimension 2 – Agility Characterisation:

The second dimension of the 4-DAT is agility characterisation. It is a collection of agile features derived from the literature review which can be measured to identify the agility of the methodology. As this project aims to develop the most beneficial practices of each methodology, dimension two will focus only on the agile properties of each method at a phase and practice level. However, to help complete the project objective of helping to overcome the identified risks from the literature review, two additional features have been added to Qumer & Sellers (Qumer& Henderson-Sellers, 2006) second dimension of 4-DAT, communication and tracking, as they are important to current day software development. Table 6 below highlights each of the agile features that will be used to determine the agility of each methods practices.

Table 6 Dimension 2 Values

Agile Feature	Description
Flexibility (FY)	Does the practice welcome change, expected or unexpected?
Speed (SD)	Does the practice produce results quickly?
Leanness (LS)	Does the practice follow the shortest time span, using simplistic, quality tools for production?
Learning (LG)	Does the practice apply gained prior knowledge to learn from the past?
Responsiveness (RS)	Does the practice respond quickly to change?
Communication (CN)	Does the practice encourage effective communication?
Tracking (TG)	Does the practice allow for tracking to take place?

The agility of the methodology practices can be measured in terms of the seven features listed in Table 6. The possible value of a feature is 0 or 1. If the feature is true during a particular practice of a method then it is awarded a 1. If the practice does not accommodate a feature then it is awarded a 0. Qumer & sellers (Qumer& Henderson-Sellers, 2006) define the degree of agility (DA) for each practice as the fraction at which the agility features are included and supported. The equation used to measure the degree of agility is:

$$DA\,(Object) = \left(\frac{1}{m}\right)\Sigma m DA\,(Object, Practice)$$

Equation 1 Dimension 2 - Calculating Agility of Practice

Table 7 below demonstrates how the agility measurement approach is implemented.

Table 7 Sample of Agile Practice Measuring (Example taken from completed analysis within this project)

	Agile Features							
XP	**FY**	**SD**	**LS**	**LG**	**RS**	**CN**	**TG**	**Total**
Pair Programming	1	1	0	1	1	1	1	6

As Table 7 illustrates each agile feature is awarded a 0 or 1 on the basis that the practice, The Planning Game, supports each feature. Once each feature is awarded a number the total of each practice is calculated. This process is then carried out for every practice within the methodology. At this point the total of every practice is accumulated and entered into the below equation to calculate the methodologies degree of agility:

$$DA = \frac{Total}{(Number\ of\ Practices \times Number\ of\ Features)}$$

Equation 2 Dimension 2 - Calculating Agility of Methodology

Through carrying out the analysis of this dimension, it will provide empirical data that will act as a measurement against which the revised analysis can later be compared. As an objective is to ensure that each methodology remains agile in its nature the data produced will enable the identification of a drop or rise in agility, ensuring rigour to the research.

Dimension 3 – Agile Values Characterisation:

The third dimension of 4-DAT is focuses on agile values characterisation. It aims to inspect the support provided by each agile value adopted by the multiple practices used by each of the two agile methods. The original tool used in the study by Qumer & sellers (Qumer& Henderson-Sellers, 2006), uses six agile values; four of which are from the agile manifesto and two of which were instigated by authors.

However, in this research project it is the intention to use completely new agile values rather than those previously used. Through study of literature it is identified that XP and Scrum each have their own set of values. As the objective of this project is to create a hybrid methodology combing aspects of XP and Scrum, the individual agile values adopted

by either method will be combined to identify if either methodology can support the values of the other. The agile values adopted by each methodology are shown below in Table 8.

Table 8 Values of Scrum and XP

Scrum Values	XP Values
Commitment	Communication
Focus	Feedback
Openness	Simplicity
Respect	Respect
Courage	Courage

As Table 8 reveals each methodology both share the value of respect and courage thus highlighting the importance of these particular values to agile software development. The values above will be combined to create one set of values against which the practices of both methodologies will be matched, Table 9 describes each value.

Table 9 Amalgamation of Scrum and XP Values

Values	Description
Commitment	Which practices value commitment to the team and to the project?
Focus	Which practices value focus to be given?
Openness	Which practices value openness among the team?
Respect	Which practices value that respect is shown within the team?
Courage	Which practices value that courage is used in the development process?
Communication	Which practices value communication and encourage it?
Feedback	Which practices value that feedback be given?
Simplicity	Which practices value simplicity to be effective?

Dimension 4 – Software Process Characterisation:

The fourth dimension is a collection of components of software process. In their study Qumer & Sellers (Qumer& Henderson-Sellers, 2006) use four different processes under the two central categories of software processes, product engineering and process management. With the product engineering process they break it down into a further three individual processes; development process, project management process and configuration control process. For this research project a review process will be added as this is an area that was identified in the literature review as being poor and therefore any investigation into this may help in overcoming this weak point. Table 10 below will list and describe each process which will be analysed.

Table 10 Software Processes

Process	Description
Development Process	Which practices are used in the main lifecycle process?
Project Management Process	Which practices are used in the overall management of the

	project?
Software Configuration Process	Which practices are used to support the configuration process?
Review Process	Which practices are helpful in ensuring a review process?
Process Management Process	Which practices cover the process that is required to manage the process itself?

3.2.3 Questionnaire

Questionnaires are regarded as being one of the most widely used and effective research methods to gain views on a particular subject (Blaxter et al., 2006). To gather data from a human perspective a questionnaire will be created using the internet site; 'http://kwiksurveys.com'. This website provides the tools to create a questionnaire which has been distributed onto the internet through a URL address for people to complete, a copy can be found in Appendix E. The purpose of the questionnaire within this research project is to provide support to the findings of the gap based analysis which will be carried out. Further to this the questionnaire may also provide new information relating to the strengths and weaknesses of each methodology thus providing a deeper understanding of what a hybrid approach should achieve as it will be more applicable to the current environment within software development.

The target audience of the questionnaire will be professionals who have experience working on projects which have adopted agile methodologies, thus making the data which it produces supportive of the current software development environment and in keeping with current trends. The questionnaire has been published on a number of professional agile methodology groups within a professional website thus enabling it to gain responses from an audience that can produce truer answers.

Having the questionnaire published on a public website for respondents to complete it is identified as being a self-completion questionnaire. This type of survey brings with it several advantages and disadvantages and the design and construction of it must be carried out with caution.

Advantages

The first advantage of the self-completion questionnaire is that it is cheap to construct and administer to the public. With very low costs it means that the worry of only being able to afford to produce a certain number of questionnaires does not exist thus leading to the second benefit.

The second advantage of using this type of questionnaire as a research method is that it can reach a large number of people in a shorter period of time. Having access to such a potentially large resource pool means that the data collected can be in depth and represent a wide variety of opinions.

Thirdly, this type of questionnaire can also be more users friendly. The absence of an interviewer will allow the participants to relax and complete the questions in their own time being able to think about their answers without feeling hurried into making a decision.

Disadvantages

Whilst there are many benefits to using a questionnaire posted on a public website there are also several drawbacks which can cause speculation as to the quality of data collected. The primary disadvantage of using an internet questionnaire is being unable to know who is responding and if the data they are giving is truthful as many users on the internet undertake alternative identities. As this questionnaire is being published on a professionally backed website this risk is reduced but there still remains an element of uncertainty as to trueness of results gathered

An additional disadvantage that can adversely affect the results of a questionnaire is its length. If the questionnaire is too in depth and contains to many questions the respondent is likely to become bored and frustrated consequently causing them to not finish or to begin responding with false information. This again can bring into question the quality of the data collected.

Questionnaire Guidelines

In order to gain the greatest benefits from using a questionnaire as a research method there are many factors, inspired by Bryman (Bryman& Bell, 2007), which will be taken into consideration when creating the questionnaire.

1. A short introduction will be given at the beginning of the questionnaire to allow the respondent to understand what purpose of the questionnaire is and how their responses will help further improve the research of this project.
2. The questions included will be short and specific in order to keep the respondents focused and to exhibit the idea that the process of answering will not take up a long period of time.
3. The layout of the questionnaire will be simplistic and professional in appearance and provide instructions throughout to make the process as simple as possible for the respondent.
4. Each question will follow the format of multiple-choice. Whilst this limits the responses for each question it is easier to measure the data once collected.
5. At the end of the questionnaire a section will be provided for the respondent to provide any additional information that they feel is related to the questionnaire.

Through following each of these steps the questionnaire constructed will provide further data to the alternative analysis techniques which will be implemented. The structure that the questionnaire will take as is described in Table 11 below.

Table 11 Questionnaire breakdown structure

QUESTION	DESCRIPTION
Question 1 – 4	These introductory questions will help to gauge the experience of the respondents and what methodologies they have been involved with using during software develop projects.
Question 5	Will help to identify which practice(s) are the most commonly implemented among Scrum users.

Questions 6 & 7	These questions act as evaluative questions to gain an understanding of how effective users of the Scrum method found each practice and the overall methodology.
Questions 8 – 10	This follows the same structure as questions 5 – 7 but from the perspective of users of the XP methodology
Questions 11 & 12	These questions aim to identify the advantages and disadvantages that users of Agile methodologies have found to exist.
Question 13	This question takes inspiration from the identified failure from the literature review and aims to identify which of the problem areas respondents have experienced.
Question 14	Allows for respondents to add any additional information which they feel may be relevant.

NOTE:
Based on their answer to question 4, logic has been applied to the questionnaire to allow for respondents to be taken to the section which regards there chosen method. For those who answer; None; they will go straight to question 11 as they can still provide an insight into the common advantages and disadvantages found within software development projects.

4.0 Presentation of results

Section 4 of the project will detail the results that were produced through completion of the analysis of both XP and Scrum. The section will primarily highlight the findings from the analysis methods implemented and provide discussion of what they represent in relation to meeting the objectives of this project. When presentation of the results gained from analysis has been completed for both original methodologies, this section will go on to illustrate the findings from the analysis carried out with regards to the revised methodologies thus providing a clear and concise medium for identifying any improvements which have been made.

4.1 Comparative Analysis

The primary method of analysis which was applied in this project was that of a comparative analysis. The practices of either methodology were analysed individually to identify how effectively they contributed to overcoming the identified failures of software development projects which were declared in the literature review (Table 2). The results presented from the comparative analysis are also related to the finding obtained from responses to the questionnaire giving a theoretical and first-hand insight into the support given to the ten failure areas. Appendix E provides a full breakdown of results from the questionnaire.

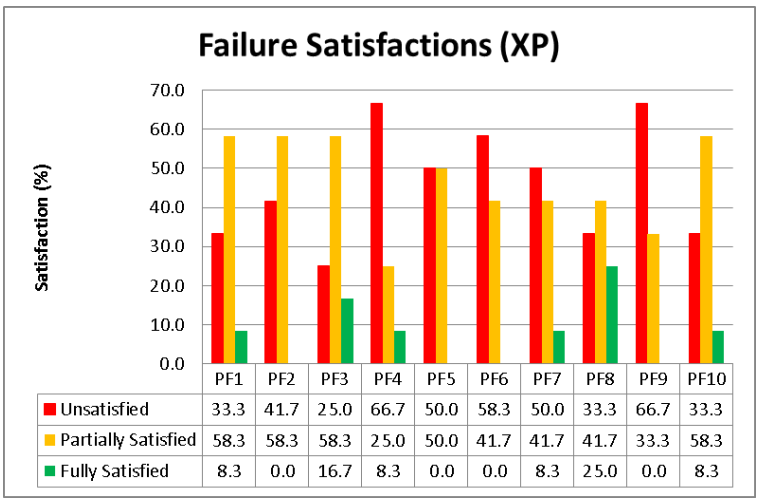
Each practice within either methodology was broken down to the specific elements that fabricated a practice. If an element within a practice could in some way contribute to helping to overcome or minimise the effect of an identified failure then it was highlighted. If all elements of a practice could contribute to aiding the solution to a failure then it was awarded a status of Fully Satisfied (FS). However, if it were only possible for some of the individual elements to help then the practice would be awarded a status as Partially Satisfied (PS) and lastly a practice in which all of its elements could not contribute in any way to the helping of overcoming a failure would be awarded an Unsatisfied (U) status.

The higher the percentage that is attributed to the status of the 'Unsatisfied' category indicates the failures which are least supported by the practices of a methodology and the elements which each contain.

4.1.1 Extreme Programming Comparative Analysis

As is exemplified in Figure 5, the analysis of each practice invoked by XP illustrates that failures PF4, PF6, PF7 and PF9 are all poorly contributed towards, each gaining a higher 'unsatisfied' percentage than that awarded to being 'satisfied'. The results gained from the comparative analysis relate to those which were found from examination of responses of the questionnaire. Whilst PF7 did was not shown to be the least supported failure from the comparative analysis, it was identified in the responses of the questionnaire to be the number one poorly support failure with 18% of respondents selecting it. The questionnaire also found that respondents believed that PF6 was an area in which much improvement was required with 13% selecting it. Whilst the results of both analysis methods do not match exactly, they do both highlight that the particular areas of PF6 and PF7 are somewhat responsible to the detrimental effect witnessed within software development projects. Interestingly, failure areas PF4 and PF9, both of which were identified to be the least

supported areas by the comparative analysis were identified by questionnaire responses to not be seen has as big a threat, receiving 4% and 9% respectively. This is evidence which, whilst contradictory, enables a greater understanding of the support provided by XP as it produces results which are theoretically founded and based on real experience. Due to these weaknesses these areas will gain priority in the revision process of the practices so as to develop a methodology which stronger.



	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Unsatisfied	33.3	41.7	25.0	66.7	50.0	58.3	50.0	33.3	66.7	33.3
Partially Satisfied	58.3	58.3	58.3	25.0	50.0	41.7	41.7	41.7	33.3	58.3
Fully Satisfied	8.3	0.0	16.7	8.3	0.0	0.0	8.3	25.0	0.0	8.3

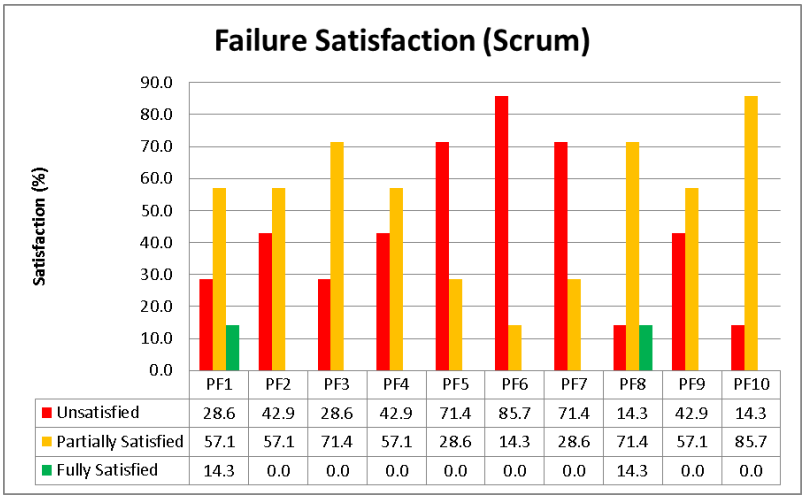
Figure 5 satisfaction results of practices against failures

Conversely, XP does contribute strong support to overcoming a delivery date which was too ambitious (PF3). This attributive strength is as a result of ethic employed by XP which commands that the development process be completed in small iterative phases therefore allowing the re-adjustment of resources and project planning initiatives.

The full breakdown of each practices contribution to a solution to overcoming the identified failures can be found in Appendix A.

4.1.2 Scrum Comparative Analysis

Analysis of the practices within Scrum provides results which follow the trend identified by the analysis of XP, as seen in Figure 6. As seen in the analysis of XP, failures which are most poorly met are PF6 and PF7, with Scrum also having the addition of PF5 for Scrum. The results here again correlate to those obtained by the questionnaire with 60% of the respondents who answered about Scrum saying that they believed PF7 to be an existing threat to Scrum based software development.



	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Unsatisfied	28.6	42.9	28.6	42.9	71.4	85.7	71.4	14.3	42.9	14.3
Partially Satisfied	57.1	57.1	71.4	57.1	28.6	14.3	28.6	71.4	57.1	85.7
Fully Satisfied	14.3	0.0	0.0	0.0	0.0	0.0	0.0	14.3	0.0	0.0

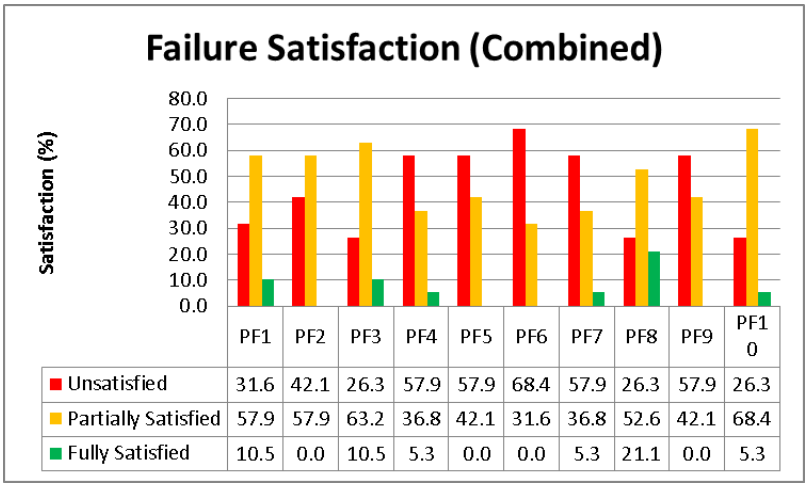
Figure 6 Scrum satisfaction results of practices against failures

Failures PF2, PF4 and PF9 also received poor results despite being awarded a higher 'satisfied' percentage than 'unsatisfied' due to the gap between the scores being minimal such as the 14% gap observed in PF2 thus meaning that the potential for the risk to exist is still significant. Conversely, failures PF3, PF8 and PF10 all score well indicating that whilst Scrum has problems integrating the aid of the customers for project scheduling, it does have the strength of adapting well when put under pressure from scheduling, a factum which is supported by 60% of respondents using Scrum in the questionnaire believing that Scrum offer the advantage of greater adaptability. A full breakdown of result relating to the comparative analysis carried out on Scrum can be found in Appendix B.

4.1.3 Combining Results within Comparative Analysis

Through the combining of results gained from the comparative analysis of XP and Scrum it provides an overall illustration of the support provided towards the ten project failures. As Figure 7 depicts, the failure areas which are positively backed by the practices from both methodologies are PF1, PF2, PF3, PF8 and PF10, a result which is backed from the overall findings gained from the questionnaire, each receiving less than 9% of votes with the exception of PF1 which gained 13% of votes. Therefore this means that these areas are lower in the prioritisation of requiring improvement as they are currently the best supported. However, it must also be noted that whilst these identified failure areas are more sustained than the other failures, they were still found to be in the top ten development project failures and so any improvements which can further enhance the reinforcement of these areas must be considered.

Alternatively, Figure 7 clearly shows which of the ten project failure areas are not well sustained by the practices of either methodology. The results here follow the trend that was witnessed amongst the individual analysis of both XP and Scrum, with PF4, PF5, PF6, PF7 and PF9 all receiving poor scores. This therefore means that when revising the practices of each methodology, it is these failures which will gain the primary attention to enable a potential for immediate improvement.



	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Unsatisfied	31.6	42.1	26.3	57.9	57.9	68.4	57.9	26.3	57.9	26.3
Partially Satisfied	57.9	57.9	63.2	36.8	42.1	31.6	36.8	52.6	42.1	68.4
Fully Satisfied	10.5	0.0	10.5	5.3	0.0	0.0	5.3	21.1	0.0	5.3

Figure 7 Combined satisfaction results of practices against failures

Table 12 summarises the prioritised list produced as a result of this method of analysis.

Table 12 Prioritised list of project failures

PRIORITY	FAILURE ID	FAILURE
1	**PF6**	Customers/Users not involved in the making of schedule estimates.
2	**PF5**	Review process at the end of each phase was not extensive enough or did not exist.
3	**PF4**	Scope changed regularly throughout the project.
4	**PF7**	Customers/Users did not make adequate time available for requirements gathering to take place.
5	**PF9**	Projects were identified as being over budget.
6	**PF8**	Projects were often behind schedule.
7	**PF2**	Highlighted risks were not continually re-assessed, controlled or managed effectively throughout the project.
8	**PF1**	The project was underestimated.
9	**PF3**	The delivery date was often too ambitious, negatively impacting the delivery process.
10	**PF10**	The development team under delivered on initial commitments.

4.2 Four-Dimensional Analysis

4.2.1 Dimension 1 – Methodology Scope Characterisation

The first dimension that is analysed as part of the 4-DAT is a quantitative comparison between the scope characteristics of XP and Scrum based on literature and academic sources. The scope characteristics which are analysed cover seven aspects and the results of which are detailed in Table 13.

Table 13 Dimension 1 Methodology Scope Characterisation

SCOPE	EXTREME PROGRAMMING	SCRUM
PROJECT SIZE	Small – medium	Small – medium but can be scaled to work with larger projects.
TEAM SIZE	2 – 12 people	8 people (multiple teams can

		exist)
TEAM STRUCTURE	Collaborative, self-organised team	Self-disciplined
DEVELOPMENT STRATEGY	Iterative	Iterative
DEVELOPMENT ENVIRONMENT	• Members code in pairs • Work in open space • Project related information displayed around workplace	
COMMUNICATION CULTURE	• Team sit together • Code and peer review in pairs • Information communicated through visual display around workplace • Stories communicate requirements in simplistic terms	• Team meet for 15 minutes daily to discuss progress • Team meet to plan each sprint • Team meet to review progress at end of each sprint
ANALYSIS STRATEGY	Team meet weekly and quarterly to discuss progress and evaluate project	Team meet daily and at the end of each 30 day sprint to discuss progress and evaluate project

Project Size

As Table 13 highlights, XP is suited to being applied to small or medium sized projects. This suitability comes from the iterative way in which XP approaches software development. If it is applied to a project which is larger in size then the potential success rate will be seen to diminish due to the longer period of time that it would take to complete and also the possibility for over complex changes in the projects requirements may materialise.

Similarly, Scrum copes best at being applied to projects which are small to medium in size. However, there is the possibility for Scrum based projects to be scaled up to a larger size, such as a project which is more geographically widespread. This is made possible by the ability for Scrum to have several teams working on the same project. However, as Schwaber & Beedle (2002) state that there are various considerations which must be taken into account if applying Scrum to a larger project. The primary point made is that parallel development should not be used from the beginning of a development project as this brings with it great risk and often has a larger negative effective. Instead, a minimal application should first be developed which goes through all layers of architecture, incorporating key requirements from the Product Backlog thus providing a platform, or common template for the other development teams to develop further

Team Size

Each methodology is designed to work effectively with smaller sized teams, thus making the team more compact and unified, avoiding ambiguity among team members. XP states that a team size ranging from two to twelve is preferred to ensure that the most benefits can be captured through implementing each of its practices (Beck, 2004).

Alternatively, Scrum believes that in order to gain the greatest benefits from each of the practices it enforces, a team of eight, just as is found in a scrum in rugby, will help achieve this. It is also mentioned that whilst a Scrum team must be eight people in size it is allowed for satellite teams to exist to help the Scrum methodology adapt and meet the requirements of a greater workload brought about by larger projects (Schwaber & Beedle, 2002).

Team Structure

A core value proposed by XP which is deemed as compulsory is teamwork. Therefore, an XP team must act as a single unit and for this to be possible managers; developers and clients are all equal in what is a collaborative team. The XP team works in a self-organising fashion focusing its attention on a problem and finding a solution to overcoming it as efficiently as possible.

In a not dissimilar way, a Scrum team must also rely on self-discipline to develop and deliver a product effectively. Unlike XP, where there is the existence of managers, a Scrum team has no direct manager to report to and therefore must be collectively self-disciplined in order to complete tasks. To aid the Scrum team the Scrum Master ensures that each of the principles and practices connected with Scrum are enforced.

Development Strategy

Each methodology employs the development strategy of rapid, iterative development, that is to say break down the project aim into smaller sub-tasks which are further refined to a size which can be worked towards within each development phase of the project, as Beck (1999) describes “a little at a time”.

Development Environment

The environment which is employed by XP teams for the development of products is a very open workspace layout. The implementation of barriers or boundaries between each team member is believed to restrict and discourage effective, constant communication from occurring. XP also believes that information relating to the project should be displayed publicly within the workplace to provide a point of quick reference for the development team.

Communication Culture

As Table 13 illustrates both XP and Scrum have several measures in place to ensure that effective communication between each of the team members and communication with project stakeholders is given the greatest possibility to exist.

XP enforces practices such as having the team sitting together to allow the opportunity for constant communication as well as the advantage of osmotic communication which was described within the literature review. This methodology also puts forth the idea of having an area of the workplace dedicated to displaying all relevant information about the project. This acts as a quick point of reference for team members to save time getting in contact with other team members and disrupting their work. Lastly, to further improve the communication potential, XP applies pair programming. This practice involves two programmers working on a single unit developing the same piece of code; one developer writing, the other constantly peer-reviewing. This practice has been found to be an effective tool in software development, lowering the number of mistakes made thus helping to reduce overall development times and budgets.

Scrum also has methods to raise the opportunity for communication to exist, allowing each team member to constantly keep up to date with information regarding the project. Each day the Scrum team will have a fifteen-minute status meeting in which they discuss what each person has completed, what they intend to do over the next day, highlighting any problems which they have identified as existing or being a potential hazard to the project. Furthermore, the Scrum team meet before and after each development phase or 'sprint' to prioritise what will be developed within the forthcoming thirty day timeframe, and also to review their performance after each phase is finished promoting team communication and learning proficiencies.

Analysis strategy

Both methodologies implement several practices to aid in the tracking of progress and the review of completed development phases.

XP's first method of analysis is the Weekly Cycle practice. This practice aims to push the XP development team to produce a piece of working software each week. This piece of software is then shown to client for review, providing an opportunity for the client to give feedback and highlight to the development team if what has been produced is unsatisfactory enabling the development team to change the product immediately. XP also adopts a Quarterly Cycle which takes place every three months. The purpose of this is to offer the client to review the developed product on a larger scale. It also provides an opportunity for the client to notify the development team of any changes to product scope that they would like to see introduced.

Conversely, the review and analysis strategy adopted by Scrum is different. At the beginning of each day the Scrum Team meet for a short update meeting. This meeting provides each of the team members the opportunity to tell the rest of the team what they have done in the last twenty four hours and what they intend to accomplish over the following day thus ensuring that everyone is made aware of what tasks are being completed. It also provides an opportunity for problems or potential risks to be highlighted so that they can be avoided or overcome as a result of collective intuition during this meeting. The Scrum Team also meet with the client before each sprint to distinguish the priority of each piece of functionality which must be developed in the development phase.

4.2.2 Dimension 2 – Methodology Agility Characterisation

The second dimension of the four part analysis focuses on the Degree of Agility (DOA) of either methodology which is measured against seven variables identified in Table 6. This analysis dimension covered two areas within each methodology, the phases and the practices.

Phases

The first part of analysis performed within this dimension was carried out to identify the DOA of the different phases, implemented by both methodologies, within a software development project. Unlike the study completed by Qumer & Sellers (2006), the phases used in this analysis were simplified to mimic that of a more traditional methodology thus helping to make the transition from a traditional based methodology to an agile method easier to it users, as one respondent of the questionnaire stated "it can be difficult to get developers to start using an agile methodology when they are so used traditional project management methodologies". Furthermore, the phases used in this analysis are the same for each methodology to allow for a simpler comparison consequently removing ambiguity.

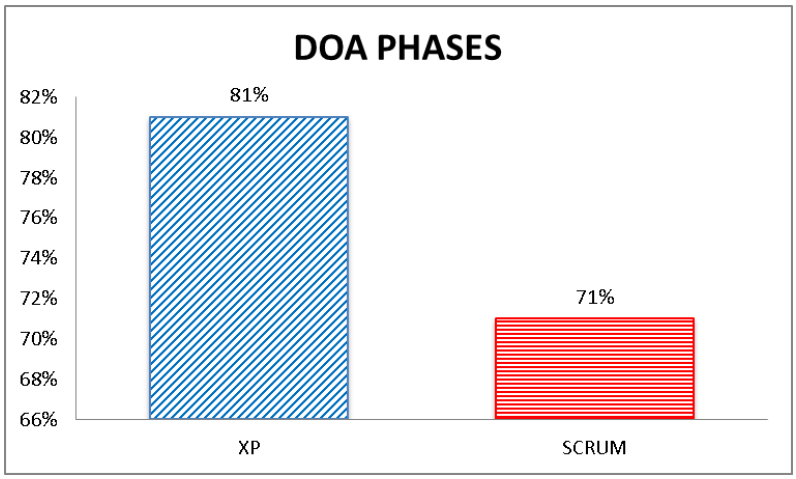


Figure 8 Degree of Agility for overall phases

Figure 8 illustrates a summary of the results gathered from analysis of the phases of each methodology, the full breakdown of analysis can be found in Appendix D. As Figure 9 identifies the total DOA that XP possesses is much greater than that of Scrum with regards to the phases of a project. The DOA is higher for XP due to the phases of pre-development and development providing much support to the seven values against which they are measured; flexibility, speed, leanness, learning, responsiveness, communication and tracking. However, the DOA of XP's phase score was lowered by its poor score awarded to the post-development phase. This poor support coincides with the information that was gathered during the literature review and that of the comparative analysis, which identified insufficient support was provided towards the review phase of software development projects.

In contrast to XP, Scrum is rewarded with a much lower score with regards to the DOA of its phases. In a similar trend to that found through the analysis of XP, the pre-development stage of Scrum was found to be the strongest phase. With practices such as the Sprint Plan Meeting and the Daily Scrum Meeting they enable the pre-development of tasks to be well

supported as there is a conscious attempt to ensure that each member of the development team understands what needs to be developed before beginning. However, the final phase, post development, scores poorly just as XP did. This poor score is due to the insufficient opportunities for review to be undertaken post development. Whilst Scrum implements a Sprint Review Meeting practice this only happens at the end of each sprint cycle, thus it is not until completion of the development period that the client can provide feedback as to the developed product.

As Table 12 from section 4.1.3 identified, the second most poorly common project failure was that of an insufficient review process being implemented. From the analysis carried out within this dimension with respect to the phases of development of both methodologies, post development was found to be equally poor thus reinforcing the point that improvements must be made to enable a stronger review process to exist.

Practices

The second section of analysis performed within dimension two, looks to measure the DOA of each of the practices included within either methodology. This method of analysis is similar to that performed by Qumer & Sellers (2006) and follows the same concept of rules. However, it must be noted that the XP methodology has undergone alteration since the completion of their study, with many of the practices being renamed and the elements adjusted. Furthermore, with the addition of two further factors for each practice to be measured against this has also contributed to changing the overall degree of agility awarded to each methodology from the original study.

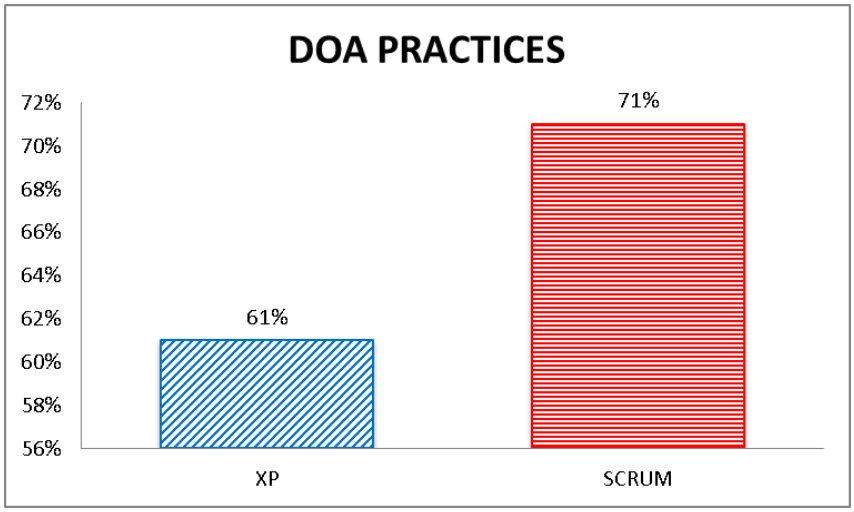


Figure 9 Degree of Agility for overall practices

Figure 9 depicts the summary of results collected from thorough analysis of the individual practices implemented within the two methodologies, a complete breakdown of the results can be found in Appendix D. As the graph shows the practices within Scrum contribute a DOA of agility which is 10% higher than that of XP.

The score awarded to XP is gained from implementing a wide variety of practices, some which boast great agile capabilities and those which are more disciplined in their characteristics. Practices such as the Informative Workplace, Pair Programming and

Continuous Integration all score highly in the analysis due to their ability to constantly change with the environment in which they are applied. Conversely, practices such as Whole Team, Energised Work and Slack each score poorly with regards to DOA. This is not due to their inability to adapt to an ever changing environment but because they are practices which; rather than execute an activity, they invoke a method of how to approach development.

The practices within Scrum exhibit a greater overall degree of agility compared to that of XP. With the exception of the Scrum Team practice, each of the remaining practices is each awarded a score of at least five. These high scoring practices illustrate that the practices of Scrum are very agile providing a solid, working platform on which to develop a stronger methodology. The weakest of the practices found to exist within Scrum with consideration to agility was the Scrum Team. Similarly to the weak practices that were identified in XP this does not mean that the practice is ineffective, only that is does not meet the values being measured within this dimension.

In summary of the analysis carried out within Dimension 2, it has been identified that XP has a greater DOA within its phases of the development cycle compared to that identified within Scrum. However, it was found that each methodology struggled in the post development phase as the review process was not agile in its nature. The poor scores witnessed correlates with the comparative analysis which was completed in this project, further proving that developing and improving the review process is of great importance. Conversely, the results produced relating to the DOA for each of the practices that are applied within each methodology illustrated that Scrum was the stronger of the two. It was found that whilst Scrum offers a greater continuity of agile potential many of the practices used in XP are very also very agile and boast great capabilities with would strengthen any software development project.

4.2.3 Dimension 3 – Agile Values Characterisation

The third dimension of analysis has the objective of identifying which practices of either methodology support the agile values of both XP and Scrum. The data illustrated within this dimension of the 4-DAT tool is qualitative and the agile values which are declared are from those identified in the review of literature.

As can be seen in Table 14 each of the agile values is all met in some way by the practices employed by either methodology. The values of respect and courage are provided with the maximum number of practices under each methodology as these values are shared by both methods and therefore are values which are incorporated into both. Additional values which are well supported by both methodologies are focus and communication, both of which are areas recorded from the questionnaire as being the biggest advantages as a result of adopting an agile methodology. This highlights that both methodologies are direct and that when a project is started using these methodologies there is a clear aim as to what must be produced.

However, the value of commitment is poorly supported by both methodologies as few practices demand that commitment is given. This could be a potential reason as to the

identified failure of projects overrunning there schedule as there is not enough force on teams to fully commit at all times resulting in possible laziness.

The value of feedback is also poorly supported by Scrum as only two practices invoke the opportunity for it to be given. This therefore, enables the conclusion that when unifying the methodologies taking inspiration from the feedback supported practices of XP will be important in creating a stronger methodology and one which will overcome the identified failure of insufficient feedback opportunities.

Table 14 Dimension 3 Analysis

VALUES	EXTREME PROGRAMMING	SCRUM
COMMITMENT	• Pair programming • Test first programming	• Sprint
FOCUS	• Sit together • Whole team • Pair programming • Weekly cycle • Quarterly cycle • Informative workplace	• Scrum master • Scrum team • Daily scrum meeting • Sprint plan meeting • Sprint review meeting
OPENNESS	• Sit Together • Pair programming • Weekly cycle • Quarterly cycle	• Scrum team • Daily scrum meeting • Sprint plan meeting • Sprint review meeting
RESPECT	ALL PRACTICES	ALL PRACTICES
COURAGE	ALL PRACTICES	ALL PRACTICES
COMMUNICATION	• Sit together • Informative workplace • Pair programming • Stories • Weekly cycle • Quarterly cycle	• Scrum master • Scrum team • Daily scrum meeting • Sprint plan meeting • Sprint review meeting • Sprint
FEEDBACK	• Weekly cycle • Quarterly cycle • Ten minute build • Continuous integration • Test first programming	• Daily scrum meeting • Scrum review meeting
SIMPLICITY	• Informative workplace	• Product backlog • Sprint

	• Energised work • Stories • Ten minute build	• Daily scrum meeting • Scrum review meeting

4.2.4 Dimension 4 – Software Process Characterisation

In the final dimension of 4-DAT the practices of XP and Scrum which provide support for the software processes are examined. Table 15 illustrates the evaluation overview of this dimension and as with Dimension 3, the information provided is qualitative and informative.

Both methodologies provide practices which support the processes of development, project management and review. However neither methodology specifies any practices which may support configuration or process management. This analysis can therefore indicate that attention must be given towards these two processes when creating a unified methodology as it will enable it to become stronger.

Table 15 Dimension 4 Analysis

PROCESS	EXTREME PROGRAMMING	SCRUM
DEVELOPMENT PROCESS	• Sit together • Whole team • Informative workplace • Energised work • Pair programming • Stories • Slack • Ten minute build • Continuous integration • Test first programming	• Product backlog • Scrum team • Daily scrum meeting • Sprint plan meeting • Sprint
PROJECT MANAGEMENT PROCESS	• Weekly cycle • Quarterly cycle	• Scrum master • Daily scrum meeting • Sprint pan meeting • Sprint review meeting
SOFTWARE CONFIGURATION PROCESS		
REVIEW PROCESS	• Informative workplace • Weekly cycle • Quarterly cycle	• Daily scrum meeting • Sprint review meeting

PROCESS MANAGEMENT PROCESS		

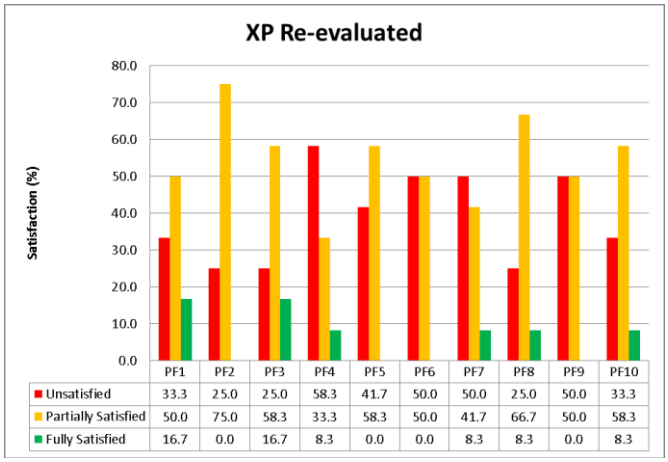
4.3 Analysis of Re-evaluated Methodologies

The following section will detail the amendments made to either methodology and discuss why the changes have been made and how they will impact the project aim of better supporting the identified common project failures. To illustrate that the changes which have been made do contribute to making the methodologies more effective they will be put through the same analysis methods thus providing comparative data which will clearly illustrate if improvements have been made.

4.3.1 Comparative Analysis of Re-evaluated Methodologies

The primary form of analysis that was completed in showing that the changes made to either methodology provided additional benefits was the completion of a comparative analysis. Many of the practices identified to be the strongest and most adaptable practices from the original analysis were altered so as to be able to provide greater support to overcoming the identified project failures. The changes which have been made to the individual practices can be found in Appendix C where a full reasoning as to why the alteration was made is given.

By subjecting these amended methodologies to the same comparative analysis as the original methodologies it provides a simplistic manner in which to identify what effect the changes have had.



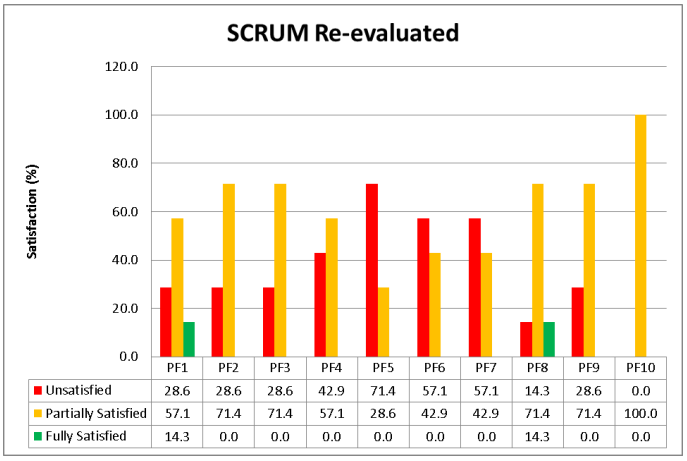
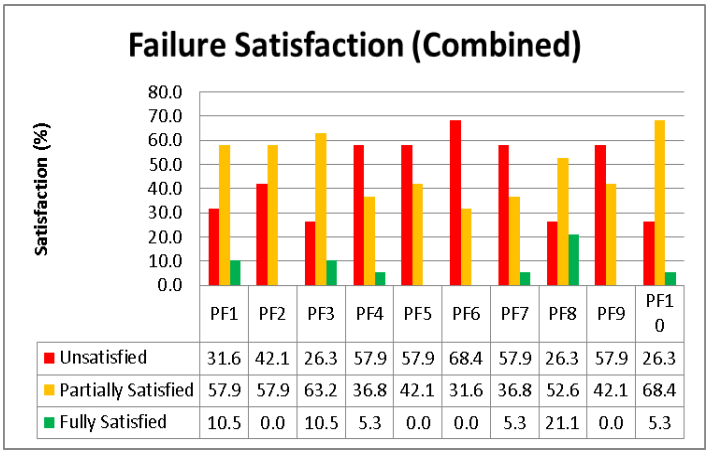


Figure 10 Comparative analysis of re-evaluated methodologies

As Figure 10 demonstrates the amendments made to the XP methodology have had an advantageous effect on the strength of support that it can provide towards each project failure, as not one failure area has gained a higher unsatisfied percentage. The greatest improvements have been witnessed in support of the customers' involvement in the making of project schedules (PF6) and the insufficient review process that existed (PF5), which has seen the unsatisfied percentage no longer outweigh the partially satisfied percentage.

In addition, analysis of Scrum against the project failures returns similar improvements. However, as Figure 10 illustrates that many of the improvements made are much more acute than those witnessed in the analysis of XP. The greatest improvement to be seen by the re-evaluated Scrum method is in its support to PF6, the customer's involvement in the making of the project schedules. Whilst the score awarded to the support of the failure still favours the category of 'Unsatisfied' of 'Partially Satisfied' the gap of 71.4% has been reduced to 14.2% therefore showing that the amendments put in place have significantly lowered the potential for this common failure to exist. Furthermore, another improvement which can be seen is the support of PF10, the under-delivery upon initial commitments, where analysis has identified that the support provided to this failure is now 100% thus largely eradicating the failure completely.

However, while Scrum has shown improvement in its support to seven of the failure areas, three failures, PF1, PF4 and PF5 have not shown enhancement towards providing better support as no change has been seen from the original analysis. Whilst this is disappointing, it is still a positive result as there is no reduction in support given; meaning that overall the methodology of Scrum now better supports the ten project failures more effectively.



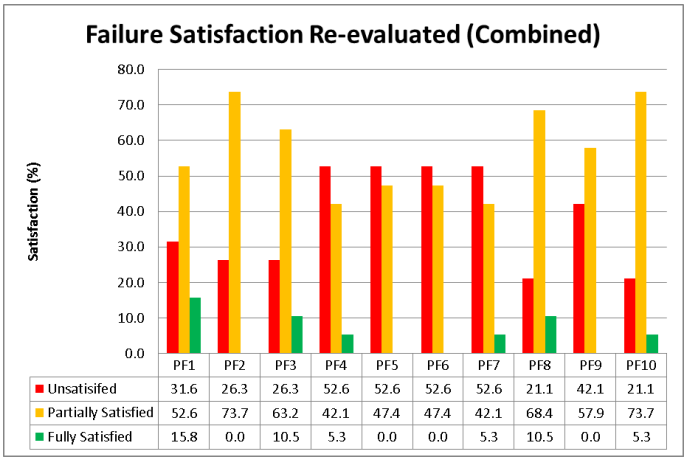


Figure 11 Comparison between original and re-evaluated

By combining the results produced from the re-evaluated analysis of XP and Scrum methodologies, produces an overview of how well each of the ten project failures are supported. As Figure 11 below highlights the change in results is significant with improvements made in the support of every project failure. As with the results of the original analysis project failures PF4 to PF6 still represent the most poorly sustained problem areas, however due to the amendments made to the practices within each methodology the gap that exists between 'Unsatisfied' and 'Partially Satisfied' is minimal thus showing that positive changes have been made. In addition, the backing to PF2, the inefficiency given to controlling and monitoring risks, has been identified to show the greatest improvement of all. This progress is vital to securing the opportunity for software development projects to operate more safely in future, as lack of attention given to this area will enlarge the potential for a project to fail.

4.3.2 Four Dimensional Analysis of Re-evaluated Methodologies

As with the original analysis the second method used to analyse the strengths and weaknesses of the methodologies is a Four Dimensional Analysis Tool (4-DAT). While each of the four dimensions will be re-analysed it is the results of Dimension 2 which will be of the most interest to this project. The reason for the enhanced interest in Dimension 2 is due the results it produces, as it identifies the Degree of Agility (DOA) that exists within the phases and the practices of each methodology. The results of such analysis are important to this research project as it will provide a clear way in which to identify if the changes made to the practices will bring about a change in the DOA.

Dimension 1 – Methodology Scope Characterisation

From the original analysis carried out, there is little that has changed within the first dimension. The changes which have come about exist within the Communication Culture of the Scrum methodology. With the additional element of setting a budget for the sprint rather than for the project as a whole, at the Sprint Plan Meeting, this offers a way in which the client and the development team can enrich their communication as the client will know how much they can afford to spend whilst the development team will have the knowledge of what the costs will be to development certain functionality. By conferring at the Sprint Plan Meeting, it enables the two stakeholders to reach an agreement thus helping to reduce the potential for the common failure of software development projects, going over budget, to exist.

As was identified in the comparative analysis, the support given to overcoming the problem of a project finishing behind schedule has seen a large increase. To further support this area, using the Informative Workplace practice of XP to not only display the current work which is required for project completion and the main risks which pose threat to the project but also to display a Gantt Chart or timeline so as that the development team cannot only identify what tasks must still be completed but by which date they must be accomplished by. Having this in place would increase the ability for the development team to measure progress and therefore ensure that they remain on target.

Dimension 2 – Methodology Agility Characterisation

As has been identified, Dimension 2 brings with it the greatest interest of the four dimensions being analysed. The reason for this greater interest is to identify if the changes which have been made affect the DOA of either methodology. As both methods are agile in their nature and characteristics any reduction in their agility would signify regression therefore meaning failure as the aim of this project was to improve upon the current methodologies.

Phases

The primary aspect that the methodologies were analysed from was how agile the phases within each were. In the original analysis it was identified that XP had a greater DOA through the three phases of pre-development, development and post-development than that recorded by Scrum. Both methods were identified as having an equally strong pre-development phase by being awarded a score of six out of seven. The result from the

analysis performed on the revised methods shows that both XP and Scrum score the same, six out seven, with regards to the first phase thus representing that no agility has been lost.

The second phase of that is investigated is that of the development phase. From the results produced in the original analysis it was found that XP had a stronger development phase scoring full points whilst Scrum scored only five from seven. Results gained from the new analysis illustrates that like the primary phase, neither method has improved nor lost any value of agility.

The last phase which is examined is that of the post-development phase. From the original analysis this phase was found to be the least agile of the three phases, each scoring four out of seven. However, results produced from analysis of the methods which included amendments do identify improvement for both methodologies. The identification of an insufficient review process within either methodology from the original analysis prioritised this area as one which would require the greatest of attention. From a large focus towards this problem it has meant that many of the changes made have been done with satisfying this aim. It is therefore the reason why the post-development phase of both, XP and Scrum, has seen an improvement with both now scoring five out of seven.

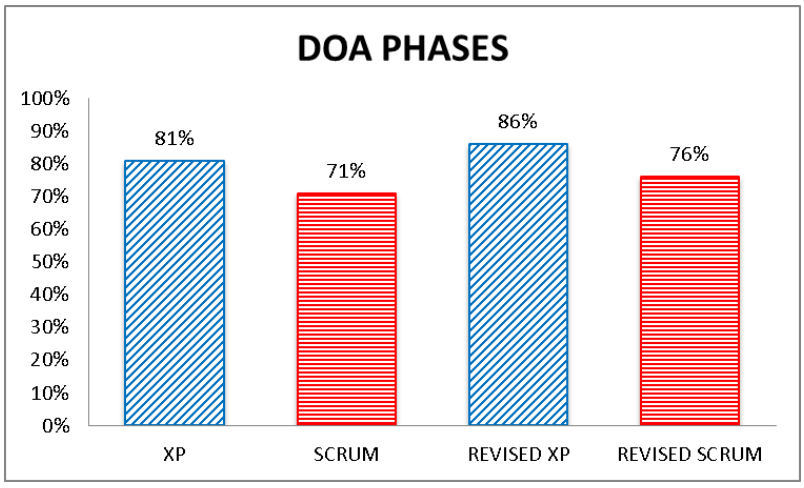


Figure 12 Degree of agility, original analysis compared with revised analysis for phases

The success in the final phase has led to both methods DOA score improving from that of the original analysis as is depicted in Figure 12 As it illustrates both methodologies have seen a 5% increase in their overall DOA over the three phases, producing the positive result that the changes which have been implemented have bettered the DOA.

Practices

The secondary aspect which either methodology were analysed was that of the DOA of the practices involved within each. The results produced from the initial analysis identified that overall the practices of Scrum were more agile than those implemented by XP. As was the objective within the analysis of the phases of each revised method, the aim of this analysis is to ensure that the DOA measured is not a reduction on original figures as this would signify reversion. Figure 13 below illustrates the overall results of the analysis carried out on the each of the revised methods and provides a direct comparison to results gained from the

original analysis. As is seen both methods have shown improvement, XP with an increase in DOA of 2% and Scrum with an increase in DOA of 9%.

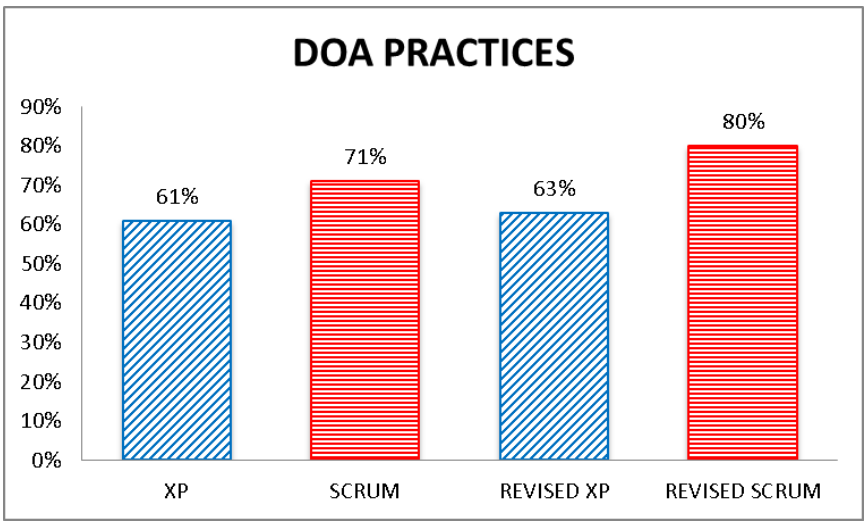


Figure 13 Degree of agility, original analysis compared with revised analysis for practices

The improvement in DOA of XP and its practices has been invoked through a change to the practice of Sit Together. With the ability of sitting near each other, developers within the team can communicate and discuss ideas rapidly thus eliminating delays in responses resulting in the development team being able to change its direction if necessary, thus improving its agility.

The practice of Continuous Integration also furthers the potential DOA measured within the XP. By continually integrating new components with the existing constituents it provides a way in which to detect faults. This prevents having to wait to the end of a development phase in order to make the necessary alterations, again improving the agility within the methodology.

Scrum has witnessed a more significant increase in its DOA as four of its practices have seen improvements in their agile scores. The Product Backlog has increased its agility as it now not only provides a way in which to record information which can be used in conjunction with development but also as a means to learn from for future projects. By identifying which aspects of the Product Backlog were effective in contributing to the project and which were ineffective, allows for future use of the practice to be implemented in such a way that it provides information that it truly required.
The Sprint and Sprint Plan Meeting have improved their DOA by improving on their tracking capabilities. By being made aware of the risks which will potentially affect the forthcoming Sprint enables the development team to prepare more effectively and identify which activities may provoke the risk to exist. Lastly the Sprint Review Meeting has seen improvement in its awarded DOA score due to greater responsiveness capabilities. Having the development team present their completed work to the client upon immediate completion of the development phase allows for client to identify anything which they feel could be done differently or added to the software to enhance it further. Ascertaining what

must be changed quickly means that the development team can begin to design plans for the next Sprint quickly helping the project remain on time.

Following from the trend witnessed with the results gained from analysis of the phases within each methodology, both XP and Scrum have shown improved agility overall therefore providing a way in which to further measure the success of amendments made to the practices as both methodologies have lost none of their agile characteristics.

Dimension 3 – Agile Values Characterisation

The third dimension of the re-analysis had the objective of identifying which of the practices within both methodologies support the agile values which are implemented by both XP and Scrum. From the analysis conducted it has been found that there are no differences to the results exist. This result therefore represents that continuity exists between the original and revised methods and that whilst none of the values has gained greater support it must also be noted that no values has seen a reduction in support.

Dimension 4 – Software Process Characterisation

The fourth and final dimension that the newly revised methodologies were analysed through aimed to identify which of the practices can support the different processes that exist within a software development project. In a similar outcome to that seen in Dimension 3, no changes were identified as existing therefore representing that no regression has happened as a result of the changes made to the practices within either methodology.

5.0 Conclusions and Discussion

The final section of this report will provide a summary of this research projects primary aim before detailing the conclusions which have made based on the results gained and illustrated in Section 4. The conclusions which are derived will then be analysed as to what they mean and how they relate to the work completed by others in similar research areas and also how the results produced relate to the projects research question. Lastly, this section will detail the limitations which were found to exist within the study, what could have been completed differently to further enhance the research and what potential areas of further research could arise as a result before drawing on a final conclusion.

5.1 Project Résumé

The aim of this project was to follow the optimism of Spence (2005), who stated that there has to be a "better way" of finding a software development methodology that can be applied to a wider spectrum of projects and which provides support to overcoming the existing failure areas. As Boehm (2002) claims, often the unification of two approaches is most beneficial to improving a modern software development project. Therefore the research question that was investigated in this project was:

"Can a hybrid agile methodology framework be created based of on the amalgamation of existing agile methodology principles, derived from XP and Scrum, and combining with disciplined principles to construct a unified methodology which can improve support given to overcoming common project failures, evaluating the results by simulating the implementation of the amended methodologies against the application of existing XP and Scrum methodologies, identifying the advantages through focused analysis?"

The research question was investigated by first identifying the most common software development project failures through a study of literature. The completion of this was accomplished by amalgamating the work of three completed software development project studies focusing on the failures to identify the ten most common causes for failure.

The project went on to analyse both XP and Scrum methodologies in relation to how the individual elements that make up each practice contained within either method provide support to the ten common failures causes. Further to this, analysis was also carried out to detect the Degree of Agility (DOA) of each of the methodologies phases and practices thus providing empirical data which the re-evaluated methods much match or improve upon.

The next stage of the project was to take the individual practices within either methodology which could be adapted or improved in some way so as that they could individually support more of the failure areas thus strengthening the support provided by the methodology which owned them.

The methodologies were then analysed again, in exactly the same way as was originally completed, in order to identify if the amendments which were made produced any significant improvements in the support which could be given to the ten project failure areas. The results produced from this analysis will be discussed more extensively in the

following section and relating how the work accomplished within this research project relates to the work completed by others in this study area.

5.2 Final Discussion of Results

The information and results that are presented in Section 4 of this report illustrate that through the enhancement of several individual practices, which are implemented by the XP and Scrum methodologies, the additional support which can be provided to overcoming the ten most common software development project failures is significantly increased, such as the 16% rise in support that is given to overcoming the common failure of highlighted risks not be continually re-assed and controlled effectively.. Furthermore, the results which are presented illustrate that by instigating these changes both methodologies gain an enriched degree of agility, an outcome which is perceived by 9% increase in the agile capabilities within the practices of Scrum. The results deliver a critical measure of the benefits that could be gained if implementation of the revised practices was initiated thus resulting in conclusion that they hold potential advantages to a software development project.

5.2.1 Relation to Research Question and Hypothesis

The findings gathered from this study have shown that by amending the practices within XP and Scrum to extend their practical capabilities has enabled either methodology to achieve greater supportive proficiencies towards the ten project failures, which can be evaluated with relation to the projects research question and hypothesis. Section 3 of this report detailed that the aim of the analysis which was to be carried out within this project was to aid the investigation of the research question through authenticating the hypothesis that an amalgamation of the practices of XP and Scrum may provide a way in which to create a stronger unified methodology which could prove to be more supportive of problematic areas that singularly, neither method can support effectively. Therefore it can be stated that the results produced in this project validate this hypothesis as the enhancing of existing practices were found to improve each methodology and in addition the combining of the practices, from both XP and Scrum, found to be more effective in the objective of providing greater support towards the top ten failures.

5.2.2 Comparative Analysis Conclusions

The primary method of analysis used to gain result in this project was the by using comparative analysis, comparing how the individual elements of the practices implemented within each methodology supported the ten project failures. The purpose of this analysis in the primary stages was to gain an understanding of the order in which the ten project failures came, beginning with the least supported and ending with the most supported. The results gained related to other studies carried out which shared a similar focus. A study completed by (Cerpa& Verner, 2009)) found that one of the most poorly completed project management tasks was the implementation of an effective review process, a result which directly correlates to the findings of the combined comparative analysis which found the same failure to be the second least supported project failure. Furthermore, this result also relates to the findings produced by the questionnaire that was carried out as it identified the insufficient review process as the second least supported failure with 13%.

The comparative analysis also identified that individually XP provided a greater support than Scrum to eight of the ten common project failures, however Scrum was found to better support the failures which dealt with falling behind original schedule estimates, PF8 and PF10 thus suggesting that Scrum was more agile when re-adjusting to time constraints, a conclusion which relates to the findings of the Four Dimensional Analysis carried out which identified the practices of Scrum to have a greater agility than found with XP.

Upon completion of the revision of the elements within the practices of XP and Scrum the comparative analysis was again applied to identify if the changes made increased the support witnessed towards the ten failure areas. The conclusions which can be derived from the second round of analysis, based on the overall results, which is the combining of results from XP and Scrum, is that with the exception of providing better support to PF3, all other failure areas observed an increase in backing. The support given to PF3 did not improve nor did it diminish in any way with amendments to the practices. It is disappointing that no improvement was able to be made; however, as it was eighth in the original priority listing it was not considered to be the most impending problem area.

The results do however show large in improvement in the support given towards overcoming PF6, originally the least support failure area. The gap identified from the original analysis was found to be 36.8% between 'Unsatisfied' and 'Partially Satisfied' yet the second analysis identifies that the gap has now fallen to just 5.2% making it the most improved failure area.

Within the individual results produced for each methodology from the completion of a second comparative analysis improvements were seen to be made in many of the failure areas. The XP methodology was seen to increase its support to all ten failures yet still unable to outweigh the 'Unsatisfied' category of PF4 and PF7 thus overall increasing its effectiveness significantly. Similarly, Scrum was seen to shown improvement in all areas; however it was still unable to totally overcome the problems produced by PF5, PF6 and PF7.

In conclusion to the comparative analysis, the results gained are very positive and highlight that it is possible to reduce each of the top ten failure areas to a more low-risk state and that with additional time to further enhance the practice within either methodology it may be possible to significantly further reduce the potential threat exemplified by each failure.

5.2.3 Four Dimensional Analysis Conclusions

The second method of analysis that was applied in this project was that of a Four Dimensional Analysis Tool (4-DAT) which was used primarily to identify the Degree of Agility (DOA) of the phases and practices within XP and Scrum. This tool was originally used in a study by Qumer & Sellers (Qumer& Henderson-Sellers, 2006), in which they too compared the differences between XP and Scrum. For the purpose of this project however the factors against which the phases and practices were measured were altered so as to be specific to this projects aim. The data gathered from Dimensions 1, 3 and 4 were all qualitative in their nature and somewhat subjective to the analysts' opinion. Therefore, whilst the data gathered is of some interest it was found to be difficult to make use of the information in an innovative manner.

The second dimension of the 4-DAT was however found to be an effective tool in applying rigour to the study. It provided a way in which to measure the agility of a particular software development methodology at a precise stage in a process, using the practices applied by a method providing a base result which would later act as a measurement against which the revised methods would be matched to identify if an increase or decrease in agility results from the changes made to the practices. The results produced from this dimension of analysis produced results which relate and followed the trend of results found by Qumer & Sellers (Qumer& Henderson-Sellers, 2006) in the original evaluation, with the phases of XP found to be 10% more agile than those of Scrum and the practices of XP to be a between 9% and 10% less agile than Scrum.

The 4-DAT was then applied again after the construction of the revised methods was complete. The analysis again focused on the results produced by the second dimension as this provided a simplistic and measurable method in which to identify if the changes which had been made had a positive or negative effect on the agile capabilities of XP and Scrum. As the results presented in Section 4 show an improvement was seen in both methodologies, both within the phases and the practices. XP observed an increase of 5% over its phases and a 2% increase with regards to it practices agility levels. In a similar trend Scrum also witnessed a 5% increase of agility in its phases but vastly improved its agile capabilities within the practices, growing by 9%.

In conclusion to the 4-DAT, the results which are produced were a helpful method in measuring that the changes which were made still allowed the methodologies to be agile in their nature. In retrospect Dimensions 1, 3 and 4, whilst providing interesting information, were not as valuable to this project as Dimension 2 and therefore would perhaps be left out in future studies which encompassed the same project aim. The outcomes produced from the second dimension however were very positive and provided a means of proving that the alteration could have constructive influence on furthering the agile capabilities of both methodologies.

5.2.4 Questionnaire Conclusions

The third and final method used to gather results which would contribute to the outcome of this project was the publication of a short questionnaire on agile specific pages within a professional networking internet site. The primary objective of this analysis method was to gain a further understanding, from the point of view of those involved in software development projects, of the extent of problems which existed and additionally how effective users found XP and Scrum to be in regards to delivering a project.

The results gathered, provided further insightful information which supported many of the findings made through the alternative methods of analysis of this project. As this project initially focused on the identification and prioritisation of common failures within software development projects, the results gained from a question which asked respondents to identify what failures they felt were most common was of particular interest as it could support the findings of the comparative analysis. The results produced found that biggest problem, with 18% of the votes, was that of the clients not providing enough time for the gathering of requirements. This result relates strongly to the findings of the comparative

analysis, which identified this failure as combined second in the priority of least supported. In addition the questionnaire provided further backing to the results of the comparative analysis by identifying that 13% of respondents found that poor client involvement in scheduling estimations and an inadequate review process were each key failures in software development projects, problems which were identified as being first and second, respectively, most poorly supported areas in the comparative analysis.

The results gained from the questionnaire recognised that those involved in agile based development projects found that the biggest disadvantage from implementing an agile approach was that of a client wanting the project to be meet a fixed budget thus producing the idea that whilst the software development community endeavour to evolve agile methodologies, clients are still requiring the safety of a more traditional, methodical approach.

In conclusion, the results generated from the responses to the questionnaire were of a strong interest to the project as they provided a manner in which to identify that the results produced from the alternative analysis methods mimicked the views of those involved with agile methodologies and software development projects.

5.3 Project Limitations and Future Work

This section has the objective of critically reviewing how successful the project was in completing the objectives of the project, set out in section 1.2.3. It is important that a completed project be critically evaluated as it identifies the strengths and weaknesses which were found and offers an opportunity to discuss how the project could have been improved had an alternative approach been taken. In addition, understanding the strengths and weaknesses of the study coupled with the results generated can provide an awareness of the future work which could be completed as a progression from this research.

Project Strengths

This study has delivered a detailed insight to the most common failures within software development projects and how the practices implemented by XP and Scrum can each offer support in overcoming these problematic areas, with the objective of evolving particular practices to further improve the potential support which can be given. The project is academically a well-founded piece which achieves the satisfaction of objectives one and two through the completion of a comprehensive review of literature. It primarily combines the findings from research of others to categorise what constitutes as failure in software development and what the probable reasons for failure are, thus satisfying objective one. Secondly, the study investigates each practice of XP and Scrum through a thorough research of relevant literature to first gain a greater understanding of each practice, and secondly, analyse how it could be possible for each practice to provide potential solutions in overcoming the problematic areas, therefore successfully completing objective two.

A comparative analysis of the practices, employed by each methodology, against the ten project failures was carried out in a concise analytical method with the aim of manufacturing a list prioritising the failure areas from least supported to that with the most support. Through using this method of analysis, the data produced was able to clearly

highlight the support which was given to the failure areas thus enabling the focus for the revision of practices to be tailored to overcoming the areas which required the most development, illustrated by Table 12 in the results section, fulfilling the requirements of the third objective.

Project Weaknesses

Whilst much of the work carried out in this project have met the original objectives set out at the beginning and the results which have been produced are supportive of the projects hypothesis there are aspects which may have enforced limitations on the research. As the project took a largely a theory based methodology to analysing the relations of between practices and the common project failure reasons it produced a substantial amount of qualitative data. Taking this approach ensured that the work was completed on a foundation of strong knowledge and understanding of either methodology. However, it may have be more beneficial to the final quality of the project had there been a greater integration and input of working experience from a practitioner who has worked in a software development environment. This would have potentially provided a more comprehensive input of primary knowledge allowing the work which was completed to possess an increased relevancy to the subject area.

Future Work

Whilst the critical evaluation of the project highlighted that limitations existed within this research project, they also propose the possibility for future work to taken on. As a weakness of this project was identified as not integrating the potential first-hand knowledge of a software development practitioner, this would create an area in which to further investigate. Whilst a study based on strong, relevant literature is good, taking the insights gained from an individual with primary experience could enhance the real-life difference made by this type of study. Agile is about making the process of software development more efficient and more effective, therefore gaining knowledge from an individual who understands the working environment would be very beneficial.

5.4 Conclusions

This project has successfully identified the most imposing causes to software development projects being deemed a failure and recognised which practices within XP and Scrum can provide a method of support in overcoming these failures. The completion of the research was achieved through a comprehensive literature review and effective analysis techniques which produced data that was both relevant and intuitive towards investigating the projects research question and hypothesis.

The data transmitted from this project could be of potential interest to those who currently employ an XP or Scrum methodology within their development environment but have encountered development issues such as those established in the research and require an insight in how to possibly overcome the issues.

This study has revealed that there are a large number of factors which can exist in the turbulent environment of software development which can cause great adverse effects to the success of a project. It has gone on to determine that each practice within XP and

Scrum can offer an individual enhancement to the development process and that their correct implementation is very beneficial. However, whilst each practice contributes to making the process of developing software a more efficient and predictable procedure there is still a large opportunity for negative impacts to materialise. Nevertheless, there are actions which can be put in place to prevent the effects of such failures being quite so influential, again increasing the predictability of the process and minimising the potential existence of risk whilst still maintaining naturism of agility. Through the development of agile methodology practices, they can be customised to provide support to a wider variety of failures or increase the focus which they offer towards a single failure area. Therefore this systematic and detailed study can conclude that the unification of principles and practice employed by XP and Scrum can contribute to a greater ability in overcoming the common failures which are often detrimental to software development projects following the original belief of Spence (2005) that there is a “better way”.

6.0 References

Ahonen, J.J. & Savolainen, P. 2010, "Software engineering projects may fail before they are started: Post-mortem analysis of five cancelled projects", *Journal of systems and software,* Vol. 83, no. 11, pp. 2175-2187.

Akingbehin, K. 2005, "A quantitative supplement to the definition of software quality", *3rd ACIS international conference on software engineering research, management and applications, SERA*

Ambler, S.W. 2002, ***Agile modeling : Effective practices for eXtreme programming and the unified process,*** John Wiley & Sons,.

Beck, K. 2004, ***Extreme programming explained : Embrace change,*** 2nd edn, Addison-Wesley,.

Beck, K. 1999, "Embracing change with extreme programming", *Computer,* Vol. 32, no. 10, pp. 70-77.

Beck, K., Highsmith, J. & Cockburn, A. 2001, *Manifesto for agile software development* [online]. Available at: http://www.agilemanifesto.org/ [Accessed October/ 3rd 2011].

Beck, K. & Boehm, B. 2003, "Agility through discipline: A debate", *Computer,* Vol. 36, no. 6, pp. 44.

Berczuk, S. 2007, "Back to basics: The role of agile principles in success with an distributed scrum team", *AGILE 2007*IEEE, , pp. 382.

Blaxter, L., Hughes, C. & Tight, M. 2006, *How to research,* Open Univ Pr,.

Boehm, B. 2002; 2002, "Get ready for agile methods, with care", *Computer,* Vol. 35, no. 1, pp. 64-64-69.

Boehm, B., Chulani, S., Verner, J. & Wong, B. 2009, "Seventh workshop on software quality", *2009 31st international conference on software engineering, ICSE 2009, may 16, 2009 - may 24*2009, IEEE Computer Society, Vancouver, BC, Canada, pp. 449.

Boehm, B. & Turner, R. 2003, "Using risk to balance agile and plan-driven methods", *Computer,* Vol. 36, no. 6, pp. 57-66.

Braithwaite, K. & Joyce, T. 2005, "XP expanded: Distributed extreme programming", *6th international conference on extreme programming and agile processes in software engineering, XP 2005, june 18, 2005 - june 23*2005, Springer Verlag, Sheffield, United kingdom, pp. 180.

Bryman, A. & Bell, E. 2007, *Business research methods,* Oxford University Press, USA,.

Cerpa, N. & Verner, J.M. 2009, "Why did your project fail?", *Communications of the ACM,* Vol. 52, no. 12, pp. 130-134.

Cockburn, A. & Williams, L. 2000, "The costs and benefits of pair programming", *eXtreme programming and flexible processes in software engineering--XP2000*, pp. 33.

Conrad, B. 2000, "Taking programming to the extreme edge", *InfoWorld,* Vol. 22, no. 30, pp. 61.

Cuthbertson, C. & Sauer, C. 2003, "The state of IT project management in the UK 2002-2003", *Computer weekly,* .

DeMarco, T. 2002, *Slack: Getting past burnout, busywork, and the myth of total efficiency,* Broadway,.

Dictionary 2012, *Dictionary.com* [online]. Available at: http://dictionary.reference.com/browse/sprint [Accessed 01/05 2012].

Dyba, T., Arisholm, E., Sjoberg, D.I.K., Hannay, J.E. & Shull, F. 2007, "Are two heads better than one? on the effectiveness of pair programming", *Software, IEEE,* Vol. 24, no. 6, pp. 12-15.

Erdogmus, H., Morisio, M. & Torchiano, M. 2005, "On the effectiveness of the test-first approach to programming", *IEEE transactions on software engineering,* Vol. 31, no. 3, pp. 226-237.

Fernandes, J.M. & Almeida, M. 2010, "Classification and comparison of agile methods", *7th international conference on the quality of information and communications technology, QUATIC 2010, september 29, 2010 - october 2*2010, IEEE Computer Society, Porto, Portugal, pp. 391.

Glass, R.L. 2005, "IT failure rates--70% or 10-15%?", *Software, IEEE,* Vol. 22, no. 3, pp. 112-111.

Glass, R.L. 2001, "Extreme programming: The good, the bad, and the bottom line", *Software, IEEE,* Vol. 18, no. 6, pp. 112-111.

Goldman, S.L., Nagel, R.N. & Preiss, K. 1995, "Agile competitors and virtual organizations", *Manufacturing review,* Vol. 8, no. 1, pp. 59-67.

Greer, D. & Hamon, Y. 2011, "Agile software development", *Software - practice and experience,* Vol. 41, no. 9, pp. 943-944.

Guntamukkala, V., Wen, H.J. & Tarn, J.M. 2006, "An empirical study of selecting software development life cycle models", *Human systems management,* Vol. 25, no. 4, pp. 265-278.

Hazzan, O. & Leron, U. 2010; 2010, "Disciplined and free-spirited: 'time-out behaviour' at the agile conference", *The journal of systems and software,* Vol. 83, no. 11, pp. 2363.

Highsmith, J. & Cockburn, A. 2001, "Agile software development: The business of innovation", *Computer,* Vol. 34, no. 9, pp. 120-122.

Hinde, S. 2005, "Why do so many major IT projects fail?", *Computer fraud and security,* Vol. 2005, no. 1, pp. 15-17.

Hughes, B. & Cotterell, M. 2006, *Software project management,* 4th edn, McGraw-Hill Education,.

Hunt, J. 2006, *Agile software construction,* Springer-Verlag New York Inc,.

Hussain, Z., Lechner, M., Milchrahm, H., Shahzad, S., Slany, W. & Umgeher, M. 2008, "Optimizing extreme programming", *Computer and communication engineering, 2008. ICCCE 2008. international conference on*, pp. 1052.

Ji, F. & Sedano, T. 2011, "Comparing extreme programming and waterfall project results", *2011 24th IEEE-CS conference on software engineering education and training, CSEE and T 2011, co-located with the 33rd international conference on software engineering, ICSE, may 22, 2011 - may 24*2011, IEEE Computer Society, Waikiki, Honolulu, HI, United states, pp. 482.

Juric, R. 2000, "Extreme programming and its development practices", *Information technology interfaces, 2000. ITI 2000. proceedings of the 22nd international conference on*, pp. 97.

Khalaf, S.J. & Al-Jedaiah, M. 2008, "Software quality and assurance in waterfall model and XP - A comparative study", *WSEAS transactions on computers,* Vol. 7, no. 12, pp. 1968-1976.

Krishnan, M.S., Mukhopadhyay, T. & Zubrow, D. 1999, "Software process models and project performance", *Information systems frontiers,* Vol. 1, no. 3, pp. 267-277.

Law, A. & Charron, R. 2005, "Effects of agile practices on social factors", *ACM SIGSOFT software engineering notes*ACM, , pp. 1.

Layman, L., Williams, L., Damian, D. & Bures, H. 2006, "Essential communication practices for extreme programming in a global software development team", *Information and software technology,* Vol. 48, no. 9, pp. 781-794.

Lindstrom, L. & Jeffries, R. 2004, "Extreme programming and agile software development methodologies", *Information systems management,* Vol. 21, no. 3, pp. 41-52.

Marcal, A.S.C., de Freitas, B.C.C., Furtado Soares, F. & Belchior, A.D. 2007, "Mapping cmmi project management process areas to scrum practices", *Software engineering workshop, 2007. SEW 2007. 31st IEEE*Ieee, , pp. 13.

Martin, A., Biddle, R. & Noble, J. 2009, "XP customer practices: A grounded theory", *2009 agile conference, AGILE 2009, august 24, 2009 - august 28*2009, IEEE Computer Society, Chicago, IL, United states, pp. 33.

Maurer, F. & Martel, S. 2002, "Extreme programming. rapid development for web-based applications", *Internet computing, IEEE,* Vol. 6, no. 1, pp. 86-90.

McDowell, C., Werner, L., Bullock, H.E. & Fernald, J. 2006, "Pair programming improves student retention, confidence, and program quality", *Communications of the ACM,* Vol. 49, no. 8, pp. 90-95.

Miller, G.A. 1994, "The magical number seven, plus or minus two: Some limits on our capacity for processing information.", *Psychological review,* Vol. 101, no. 2, pp. 343.

Millett, S., Blankenship, J. & Bussa, M. 2011, *Pro agile. NET development with SCRUM,* Apress,.

Ming Huo, Verner, J., Liming Zhu & Babar, M.A. 2004, "Software quality and agile methods", *Computer software and applications conference, 2004. COMPSAC 2004. proceedings of the 28th annual international*, pp. 520.

Najafi, M. & Toyoshiba, L. 2008, "Two case studies of user experience design and agile development", *Agile, 2008. AGILE'08. conference*Ieee, , pp. 531.

Naoum, S. 2006, ***Dissertation research and writing for construction students*** , 2nd edn, Butterworth-Heinemann Ltd,.

Nayoung Hong, Junbeom Yoo & Sungdeok Cha 2010, "Customization of scrum methodology for outsourced E-commerce projects", *Software engineering conference (APSEC), 2010 17th asia pacific*, pp. 310.

Oates, B.J. 2006, *Researching information systems and computing,* Sage Publications Ltd,.

Ota, M. 2010, "Scrum in research", *7th international conference on cooperative design, visualization, and engineering, CDVE 2010, september 19, 2010 - september 22*2010, Springer Verlag, Calvia, Mallorca, Spain, pp. 109.

Oxford Dictionary 2011, *Oxford dictionaries* [online]. Available at: http://english.oxforddictionaries.com/definition/agile [Accessed October, 13 2011].

Paasivaara, M., Durasiewicz, S. & Lassenius, C. 2008, "Using scrum in a globally distributed project: A case study", *Software process: Improvement and practice,* Vol. 13, no. 6, pp. 527-544.

Paulk, M.C. 2001, "Extreme programming from a CMM perspective", *Software, IEEE,* Vol. 18, no. 6, pp. 19-26.

Qumer, A. & Henderson-Sellers, B. 2006, "Measuring agility and adoptability of agile methods: A 4-dimensional analytical tool", *Procs. IADIS international conference applied computing 2006*, pp. 503.

Rakitin, S.R. 2001, "Manifesto elicits cynicism [1]", *Computer,* Vol. 34, no. 12, pp. 4-4.

Rasmussen, J. 2003, "Introducing XP into greenfield projects: Lessons learned", *Software, IEEE,* Vol. 20, no. 3, pp. 21-28.

Reel, J.S. 1999, "Critical success factors in software projects", *Software, IEEE,* Vol. 16, no. 3, pp. 18-23.

Reifer, D.J. 2002, "How good are agile methods?", *Software, IEEE,* Vol. 19, no. 4, pp. 16-18.

Rising, L. & Janoff, N.S. 2000, "Scrum software development process for small teams", *IEEE software,* Vol. 17, no. 4, pp. 26-32.

Rong, G., Shao, D. & Zhang, H. 2010, "SCRUM-PSP: Embracing process agility and discipline", *Software engineering conference (APSEC), 2010 17th asia pacific*IEEE, , pp. 316.

Santos, R., Flentge, F., Begin, M. & Navarro, V. 2011, "Agile technical management of industrial contracts: Scrum development of ground segment software at the european space agency", *12th international conference on agile processes in software engineering and extreme programming, XP 2011, may 10, 2011 - may 13*2011, Springer Verlag, Madrid, Spain, pp. 290.

Sauer, C. 1993, *Why information systems fail: A case study approach,* 1st edn, Alfred Walker Ltd Publishers, Oxfordshire.

Schwaber, K. 1995, "Scrum development process", *OOPSLA business object design and implementation workshop*Austin, TX, , pp. 10.

Schwaber, K. & Beedle, M. 2002, *Agile software development with scrum,* Prentice Hall PTR Upper Saddle River^ eNJ NJ,.

Sharp, H., Robinson, H., Segal, J. & Furniss, D. 2006, "The role of story cards and the wall in XP teams: A distributed cognition perspective", *AGILE conference, 2006, july 23, 2006 - july 28*2006, Inst. of Elec. and Elec. Eng. Computer Society, Minneapolis, MN, United states, pp. 65.

Shore, J. & Warden, S. 2007, *The art of agile development,* O'Reilly,.

Siebra, C.A., Filho, M.S.A., Silva, F.Q.B. & Santos, A.L.M. 2008, "Deciphering extreme programming practices for innovation process management", *4th IEEE international conference on management of innovation and technology, ICMIT, september 21, 2008 - september 24*2008, Inst. of Elec. and Elec. Eng. Computer Society, Bangkok, Thailand, pp. 1292.

Spence, J.W. 2005, "There has to be a better way!", *AGILE conference 2005, july 24, 2005 - july 29*2005, Inst. of Elec. and Elec. Eng. Computer Society, Denver, CO, United states, pp. 272.

Sutherland, J. 2005, "Future of scrum: Parallel pipelining of sprints in complex projects", *Agile conference, 2005. proceedings*, pp. 90.

Sutherland, J., Downey, S. & Granvik, B. 2009, "Shock therapy a bootstrap for hyper-productive scrum", *2009 agile conference, AGILE 2009, august 24, 2009 - august 28*2009, IEEE Computer Society, Chicago, IL, United states, pp. 69.

Sutherland, J., Viktorov, A., Blount, J. & Puntikov, N. 2007, "Distributed scrum: Agile project management with outsourced development teams", *40th annual hawaii international conference on system sciences 2007, HICSS'07, january 3, 2007 - january 6*2007, Inst. of Elec. and Elec. Eng. Computer Society, Big Island, HI, United states.

Takeuchi, H. & Nonaka, I. 1986, "The new new product development game", *Harvard business review,* Vol. 64, no. 1, pp. 137-146.

Tuomikoski, J. & Tervonen, I. 2009, "Absorbing software testing into the scrum method", *10th international conference on product-focused software process improvement, PROFES 2009, june 15, 2009 - june 17*2009, Springer Verlag, Oulu, Finland, pp. 199.

Turk, D., France, R. & Rumpe, B. 2005, "Assumptions underlying agile software-development processes", *Journal of database management,* Vol. 16, no. 4, pp. 62-87.

VersionOne. 2010, *State of agile survey 2010.* (5th), Version One,.

Vijayasarathy, L. & Turk, D. 2011, "Drivers of agile software development use: Dialectic interplay between benefits and hindrances", .

Whitworth, E. 2008, "Experience report: The social nature of agile teams", *Agile 2008 conference*Aug 4 - 8 2008, Institute of Electrical and Electronics Engineers Computer Society, Toronto, ON, Canada, pp. 429.

Williams, L.A. & Kessler, R.R. 2000, "All I really to know P about programmi I learned kindergar", *Communications of the ACM,* Vol. 43, no. 5, pp. 108-114.

Winston W. Royce 1970, "Managing the development of large software systems", Vol. 26, no. August, pp. 1-2;3;4;5;6;7;8;9.

Yi, L. 2011, "Manager as scrum master", *2011 agile conference, agile 2011, august 8, 2011 - august 12*2011, IEEE Computer Society, Salt Lake City, UT, United states, pp. 151.

Yourdon, E. 1997, ***Death march: The complete software developer's guide to surviving 'mission impossible' projects,*** Prentice Hall PTR,.

7.0 Bibliography

Abrahamsson, P., Warsta, J., Siponen, M.T. & Ronkainen, J. 2003, "New directions on agile methods: A comparative analysis", *Software engineering, 2003. proceedings. 25th international conference on*, pp. 244.

Baskerville, R., Ramesh, B., Levine, L. & Pries-Heje, J. 2006, "High-speed software development practices: What works, what doesn't", *IT professional,* Vol. 8, no. 4, pp. 29-36.

Biggs, M. 2000, "Technology won't end project failures; communication is key", *InfoWorld,* Vol. 22, no. 5, pp. 70.

Bouma, G.D. 1995, *A handbook of social science research,* Oxford University Press,.

Charette, R.N. 2005, "Why software fails [software failure]", *Spectrum, IEEE,* Vol. 42, no. 9, pp. 42-49.

Copeland, L. 2001, "Caterpillar digs into agile development methods", *Computerworld, december,* .

Cowan, C.L. 2011, "When the VP is a scrum master, you hit the ground running", *2011 agile conference, agile 2011, august 8, 2011 - august 12*2011, IEEE Computer Society, Salt Lake City, UT, United states, pp. 279.

ft n il Ionel, N. "CRITICAL ANALYSYS OF THE SCRUM PROJECT MANAGEMENT METHODOLOGY", .

Janzen, D. & Saiedian, H. 2005, "Test-driven development concepts, taxonomy, and future direction", *Computer,* Vol. 38, no. 9, pp. 43-50.

Layman, L., Williams, L. & Cunningham, L. 2006, "Motivations and measurements in an agile case study", *Journal of systems architecture,* Vol. 52, no. 11, pp. 654-667.

Sliwa, C. 2002, "Users warm up to agile programming", *Computer world,* Vol. 36, no. 12, pp. 8-8.

Willoughby, M. 2005, "Coder be agile, coder be quick", *Computerworld,* Vol. 39, no. 30, pp. 36-36.

Wood, W.A. & Kleb, W.L. 2003, "Exploring XP for scientific research", *Software, IEEE,* Vol. 20, no. 3, pp. 30-36.

APPENDIX A – XP Failure Analysis

Sit Together

The elements of the Sit Together (ST) practice provides strong support towards the common project failure of constant changing of project scope (PF4). By promoting that the development team work closely in an open environment boosts the generation of effective communication. This therefore allows team members to vocalise any changes to the development scope quickly. This promotion of communication can also enable the team to alert others to any risks which may materialise and construct a quick solution to resolve the problem, helping to control the potential effects that a risk could cause, thus providing backing to PF2.

Through encouraging constant communication be upheld this will boost the ability for the project team to brainstorm solutions to any problems that may face an individual enabling a more efficient development process and also providing a method to involve each team member more substantially in the project development.

SIT TOGETHER	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Refinement of client requirements	•						•			•
Continuous communication enforced	•	•		•	•		•			
Close working environment				•						
SATISFACTION CRITERIA	**PS**	**PS**	**U**	**PS**	**PS**	**U**	**PS**	**U**	**U**	**PS**

Whole Team

As is visible from the table below the elements of the Whole Team (WT) practice promote the implementation of greater communication among all project stakeholders. It is due to this reason that the identified project failures, PF6 and PF7, which involved cooperation between the development team and the client, are strongly supported through these elements.

Ensuring that the different stakeholder groups involved with the project work in unison rather than individually will help to enable a product to be produced which meets the requirements of the client and which also involves a simplistic development process, strengthening efficiency.

WHOLE TEAM	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Structured Communication		•			•					
Promotion of client-to-team communication						•	•			
SATISFACTION CRITERIA	**U**	**PS**	**U**	**U**	**PS**	**PS**	**PS**	**U**	**U**	**U**

Informative Workplace

The elements which combine to make up the practice of the Informative Workplace (IW) provide a large amount of support to many of the identified project failure causes. Failure areas such as frequently changing scope (PF4) are well backed by the elements of IW as the information can be displayed in a simplistic and easy to see manner within the work area. This allows each member of the development team to identify what changes have been made without having to constantly ask peers, therefore improving development efficiency more.

Having an area of the workplace which projects an overview of the project and details in a quick reference manner can enable the development team members to make sure that what they are working on is required. By not only displaying information relating to the scope of the project but also to project risks, constraints and time management factors the implementation of IW could help the practice to provide stronger support to project underestimation and scheduling failures, PF1 and PF6 respectively.

INFORMATIVE WORKPLACE	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Requirements posted on wall	•			•			•			
Updates of requirements posted as they change				•				•		•
Updates of projects risks posted on wall		•								
Information relating to project restraints posted on wall		•	•	•				•	•	
SATISFACTION CRITERIA	**PS**	**PS**	**PS**	**PS**	**U**	**U**	**PS**	**PS**	**PS**	**PS**

Energised Work

The practice of Energised Work (EW) works to support the belief that by overworking team members the efficiency of work completed drops. Therefore, by setting shorter periods of work time and allowing for breaks in development which help to lower stress and tiredness, meaning that team members works faster. Through this method of working only PF3 can be supported.

ENERGISED WORK	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Project members work no more than a 40hr week			•							
Take breaks when they feel necessary			•							
SATISFACTION CRITERIA	U	U	FS	U	U	U	U	U	U	U

Pair Programming

Pair Programming (PP) is a defining practice of the XP methodology. The elements which form the practice each provide support in some way to the identified project failures, however none are supported strongly.

Working closely with another member of the development team can provide a way for the programming which is completed to be more faultless as the constant peer review will flag any errors which may become apparent thus allowing the pair to collectively brainstorm and create a solution which will overcome the problem. This ability would mean that problematic areas such as highlighting and controlling risks (PF2) would become simpler.

PAIR PROGRAMMING	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Motivate each other to focus			•	•						
Clarify Ideas	•						•			
Working software produced quickly								•		•
Create solutions together to overcome problematic areas										
Developer Accountability										
Pair Support			•		•					
SATISFACTION CRITERIA	PS	U	PS	PS	PS	U	PS	PS	U	PS

Stories

The practice of Stories is a simplistic practice and consists of only one element, that the requirements gathered be documented in the form of small stories thus providing a simplistic and context specific way of understanding what is meant by each requirement given by the client. Due to this idea the practice supports only the failures which deal with requirements. The changing scope (PF4) is supported as the new requirements gathered are documented easily meaning that the development team can avoid wasting valuable time asking the client what exactly is meant. PF7, failure caused by poor requirement gathering on the clients' behalf, is supported by this practice because if the requirement is not documented in the correct format it will not be developed thus forcing the client to be involved.

STORIES	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Requirements documented as small stories				•			•			
SATISFACTION CRITERIA	U	U	U	FS	U	U	FS	U	U	U

Weekly Cycle

Element one of the Weekly Cycle (WC) provides support to only a single project failure, PF8. By aiming to produce a piece of working software each week this can enable the development team to measure their progress effectively and identify if their position with regards to the pre-determined development schedule. If it is noted that the team is falling behind schedule then a time management re-evaluation can take place.

The second element of this practice ensures that each task included within the project is organised by priority ensuring that the functionality which the client feels to be most important will be completed first. This element also helps to support PF8, meaning that it is fully satisfied by this XP practice. The second element also helps to give backing to PF10, the under delivering upon initial commitments. By having the client prioritise the functionality it ensures that they know what to expect to have been completed by the end of each development phase.

WEEKLY CYCLE	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Produce a piece of working software each week								•		
Develop according to priority of tasks	•		•			•		•		•
SATISFACTION CRITERIA	PS	U	PS	U	U	PS	U	FS	U	PS

Quarterly Cycle

The Quarterly Cycle (QC) provides much of the same support as that shown by the WC. However, the QC also provides an opportunity for greater review of project progress and issues relating to the project. The continual surveillance and controlling of highlighted risks benefits from this practice as they can be assessed over longer periods of time, enabling the team to construct stronger ways of governing the risks.

Project progress and budget issues (PF8 and PF9) are also well supported by the elements of this practice due to the more comprehensive review process which can take place through this practice.

QUARTERLY CYCLE	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Feedback given and incorporated into the next iteration		•			•					
Progress checked against initial commitments								•	•	•
Develop according to priority od tasks	•		•			•		•		•
Project budget checked against budget plan									•	
SATISFACTION CRITERIA	PS	PS	PS	U	PS	PS	U	PS	PS	PS

Slack

Slack provides the development team with a safety net of time should the development of a piece of functionality run over the allotted time that was originally designated for it. Knowing that there is a buffer period can help to lower the stress of team members, resulting in the work completed being of a higher quality and more error free.

Due to the nature of this practice is helps support the project failure areas which are linked to scheduling and development time difficulties.

SLACK	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Buffer period set aside for over development time	●		●					●		●
SATISFACTION CRITERIA	FS	U	FS	U	U	U	U	FS	U	FS

Ten-Minute Build

The Ten-Minute Build (TMB) provides the development team with the opportunity to run automated system builds which help enhance the time taken during development. As the table below shows the TMB offers support to half of the failure areas, with the most given to PF8. It provides this support by dictating that the build time of each component must be ten or less minutes. This quick build time provides rapid feedback that allows the development team to gain instant feedback as to the existence of any errors making the process of development altogether faster helping to overcome failures in which time is a large factor.

TEN MINUTE BUILD	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Automated system build			●					●		
Automated run at system-level										
Simplistic build process										
Early feedback provided		●			●			●		●
SATISFACTION CRITERIA	U	PS	PS	U	PS	U	U	PS	U	PS

Continuous Integration

The practice of Continuous Integration (CI) states that continually integrating and testing new software with existing software must happen enabling the idea of building the software a little at time rather than all at once as the potential number of existing errors would be too substantial to overcome. This practice supports all but three of the ten project failures making it a very positive practice. PF8 is 'Fully Satisfied' as both elements help to ensure that by developing the product in little sections will ensure that a large refactoring task at the end is avoided therefore helping the development meet time constraints.

CI also helps the identification of risks (PF2) early on thus allowing the developers to make alterations quickly without disrupting the entire development process.

CONTINOUS INTEGRATION	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Progress measurable	●		●			●		●		●
Faults identified immediately		●						●	●	
SATISFACTION CRITERIA	PS	PS	PS	U	U	PS	U	PS	PS	PS

Test-First Programming

By creating a test which is based upon the requirements that the software is expected to meet before the creation of the code and its implementation provides a method in which to ensure that the software created meets the clients' demands.

As the table below highlights the Test-First Programming practice strongly supports the failure of controlling and managing highlighted risks (PF2). This is achieved by the early feedback that it provides and the detection of errors that it highlights to the user, providing the developer with the knowledge that the software created does not meet the requirements of the client.

TEST FIRST PROGRAMMING	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Testing Before Coding		●			●					
Early Feedback Received	●	●						●		
Focus On Client Needs by Testing First		●	●							
Progress Shown	●					●	●	●		
Errors Detected Early		●							●	
SATISFACTION CRITERIA	PS	PS	PS	U	PS	PS	PS	PS	PS	U

APPENDIX B – Scrum Failure Analysis

Scrum Master

As the diagram below depicts, the Scrum Master (SM) poorly supports the ten identified failure areas. The first element involved within this practice does contribute in some way to the project failures of 8, 9 and 10 through the SM ensuring that each team member provides a short update at the Scrum Meeting enabling information relating to the status of the project being provided.

Element two of this practice only supports project failure, having the scope changing regularly. The SM can update the team with any changes to the scope at the daily meetings, however during the sprint period any new scope will not be able to be implemented but it can still help the development to have an awareness of a future change, thus allowing them to prepare more easily for it.

Element 3 provides no support to any of the ten identified project failures. Nonetheless, this is still a beneficial component to the practice as ensuring that the principles of the methodology are upheld, allows the methodology to be as effective as possible in the development of software.

SCRUM MASTER	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Controls scrum meeting		•						•	•	•
Brings attention to project changes				•						
Upholds scrum principles										
SATISFACTION CRITERIA	U	PS	U	PS	U	U	U	PS	PS	PS

Product Backlog

The Product Backlog (PB) can be seen, below, to contribute well to the support of project failures 1, 4 and 7 as was initially identified after the literature review. The PB provides a way in which all the requirements of the product, determined by the client, can be recorded and also prioritised giving the development team a clear understanding as to the what they are expected to do within each development period.

Project failure 7 is also well supported by the elements within this practice as each forces the client to be more involved in the process of requirements gathering as anything not included within this document will not be developed.

This practice could become yet more beneficial in its contribution to overcoming identified failures by not only recording the functional requirements of the end product but also recording information relating to general project requirements such as the budget and timescales expected of the project.

PRODUCT BACKLOG	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Every requirement/ piece of functionality recorded	•			•			•	•	•	
Made available to all stakeholders							•			
Kept up to date	•	•		•						
Requirements formatted as stories							•			
SATISFACTION CRITERIA	PS	PS	U	PS	U	U	PS	PS	PS	U

Scrum Team

The Scrum Team (ST) practice provides very little support to any of the identified project failure from the literature review. The first element of the practice can be seen to provide backing to Project Failure 3 and 10 as the disciplined nature of a ST ensures that, with the guidance of the SM, the team will communicate any issues among themselves and decide upon the best course of action should the find that the set delivery date is not achievable.

The second element of this practice does not deliver any backing to the project failures identified but it is still an element which can be effective in the development process. Having a small team helps to boost communication and allows each team member to interact with each other rather than a large development team where they may not have the opportunity to communicate as freely.

SCRUM TEAM	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Self-organising and disciplined Small in size			●							●
SATISFACTION CRITERIA	U	U	PS	U	U	U	U	U	U	PS

Daily Scrum Meeting

The Daily Scrum Meeting (DSM) is one of the strongest practices of Scrum with regards to supporting the identified failure areas. Constant provision of what each team member is working on brought about by element 1 of the practice helps to ensure that no duplication of development occurs therefore maximising the resource efficiency.

The third element is also very effective for a modern software development environment as factors relating to the project can change quickly and often. Having a meeting everyday enables each team member to keep up to date with what is going on with the project again helping to maximise the resources available.

In addition to bringing up immediate or current risks, the identification of problems relating to schedule targets will allow for this practice to provide more backing to the management of project resources helping to avoid finishing behind schedule and also re-evaluating development so as to avoid setting target deadlines which are too ambitious.

DAILY SCRUM MEETING	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Attention brought to current days tasks	●							●		
Current risks/ problems identified		●						●		
It is everyday			●		●			●	●	●
SATISFACTION CRITERIA	PS	PS	PS	U	PS	U	U	FS	PS	PS

Sprint Plan Meeting

The elements that exist within this practice contribute softly to a number of the project failures within software development. Element one makes available a way in which to get the client more involved in process such as requirements gathering and schedule planning helping to make sure that both client and development are understanding of the others needs and abilities.

By prioritising the requirements for the forthcoming sprint at this meeting, it assists the development team to plan how they will complete the set tasks within the development period. As was discussed in the literature review there is no definitive project manager within Scrum and therefore the common duty of delegating tasks is something which the team must decide upon themselves. By understanding who is working on specific tasks during the development period helps make the development process more efficient.

Risk awareness is an element that could be added to this practice to ensure that project failure two is better supported. Whilst the DSM provides short updates as to risk status this meeting can allow for the team to analyse the risks in more depth, potentially lowering their possible effects. Furthermore, to overcome the common problem of projects going over budget, setting a budget for each sprint may prove to be more cost effective rather than setting a budget for the project as a whole as client and development team would be able to provide more accurate short term cost evaluations.

SPRINT PLAN MEETING	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Client opportunity to provide input				●		●	●			
Prioritise requirements to be developed	●							●		●
Delegate tasks			●							
SATISFACTION CRITERIA	**PS**	**U**	**PS**	**PS**	**U**	**PS**	**PS**	**PS**	**U**	**PS**

Sprint

The Sprint, as has been discussed, is a short period of time which involves rapid development taking place. For this reason the development of functionality is easily calculated as development times are time boxed to thirty days. This aids the creation of project schedules and avoids the potential for an overly ambitious deadline to be set.

The sprint is also supportive of project underestimation (PF1) and under delivering on functionality (PF10). By contributing to each of these failure areas the development team will be under less stress to develop more, whilst the client will know what to expect from each development phase as they were responsible for prioritising the functionality due for development within the sprint.

SPRINT	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
30 day development period			●					●		
Develop only set tasks	●									●
Requirements do not change			●	●						
SATISFACTION CRITERIA	**PS**	**U**	**PS**	**PS**	**U**	**U**	**U**	**PS**	**U**	**PS**

Sprint Review Meeting

The Sprint Review Meeting (SRM) supports a number of the identified project failures well, particularly project underestimation (PF1) as each element allows the development team and the clients to re-evaluate the status of the project and adjust accordingly.

Whilst this review process takes place at the end of each Sprint, inadequate review process (PF5) is not fully met due to thirty day period between reviews. A review process occur more often to provide more extensive review capabilities as the environment of a software

development changes so often and is so dynamic that a lot of change can occur in a thirty day period. The re-evaluation of this practice will aim to accommodate this, in order to make it stronger and more effective.

SCRUM REVIEW MEETING	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Stakeholders review developed product	●				●			●		●
Team review performance	●	●	●							
Project aspects re-evaluated	●							●	●	●
SATISFACTION CRITERIA	FS	PS	PS	U	PS	U	U	PS	PS	PS

APPENDIX C – Revised Practices Failure Analysis

Appendix C describes the changes that have been implemented to the practices chosen for revision and the effects that they bring towards overcoming the ten project failures. Elements which have been altered from the original analysis are highlighted in blue, whilst new elements which have been introduced are highlighted in green.

Sit Together

The elements which together make up the practice of Sit Together (ST) were identified as one practice which offered the potential to support a solution in overcoming additional failures than those identified in original comparative analysis.

To further extend the potential capabilities of ST the additional element of 'Ability to discuss project quickly' has been added. This element has been derived from further analysis of literature as Layman (Layman et al., 2006) describes that communication, particularly, of an informal structure is of key importance to success in a software development team. Therefore, the new element accommodates this essential strength enabling the development team to confer project topics quickly.

As a result of this new element, the existing element of 'Continuous communication enforced' gains greater capabilities towards supporting other failure areas than originally identified. For example, the support towards PF7, inadequacy of requirements gathering, is now achieved as the team members can communicate with each other when an individual is unsure as to the full meaning of a gathered requirement. If the team cannot make an understanding then it identifies to them that they must go back and meet with the client to gain a stronger understanding.

SIT TOGETHER	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Refinement of client requirements	•						•			•
Continuous communication enforced	•	•		•	•		•			
Close working environment				•						
Ability to discuss project quickly		•						•		
SATISFACTION CRITERIA	**PS**	**PS**	**U**	**PS**	**PS**	**U**	**PS**	**PS**	**U**	**PS**

Informative Workplace

The first element within the revised practice of the Informative Workplace (IW) is the combination of three elements taken from the original analysis. The purpose of this is to make the element more clear and concise, rather than bring an additional benefit thus having the aim of reducing the opportunity for the development team members to not display information which they believe is not necessary but which may actually be of great use during the project process.

The second element of IW is a new element to update the potential ability of this practice. Shore &Warden (Shore& Warden, 2007) state the purpose of this practice is to transmit information which is essential to the project and when coupling this with the identified failure that projects often finish behind schedule (PF8), then it would make a great deal of

sense to ensure that not only are the tasks which are required to be completed displayed but also a time management system to enable the development team to understand their progress.
The last element which has been included in the revised practice is that the display should be updated each day. The reason for this is that software development projects are constantly changing and therefore it must be made sure that the development team have access to the most up to date information relating to the project in order to allow them to complete the project successfully.

INFORMATIVE WORKPLACE	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
All present risks/ requirements/ constraints displayed on wall	•	•		•			•		•	•
Time line/ Gantt chart displayed	•		•			•		•		
Updated daily	•	•		•						•
SATISFACTION CRITERIA	FS	PS	PS	PS	U	PS	PS	PS	PS	PS

Pair Programming
The practice of Pair Programming (PP) has seen the amendment of one new element and the introduction of another. The Collective Brainstorming is an alteration on the existing element of creating solutions together within the pair. Whilst the new element carries on the same concept, of more minds are better than one, this it also pushes the idea that the 'pair' can go to other developers within the project team to find a solution if they cannot create one which is effective enough. This allows the team as a whole to be involved and enables greater effective communication among the team as they are more aware of the work being completed.

The introductory element of, peer review, equates to combination of the original elements of 'Developer Accountability' and 'Pair Support'. By merging these elements the pair can review the others work in real time providing instant feedback reducing the number of errors found to exist within the code. This method has found to produce results which illustrate that overall errors within the code created are fewer (Dyba et al., 2007) thus reducing time taken to fix errors which in turn drives down the cost of development (PF9). However, it is stated that this can also be subjective to the participants' ability and areas of expertise and so whilst it brings many benefits it must be implemented with care and discipline to avoid adverse consequences.

PAIR PROGRAMMING	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Motivate each other to focus			•	•						
Clarify ideas	•						•			
Working software produced quickly								•		•
Collectively brainstorm		•		•				•		
Constant peer review			•		•				•	
SATISFACTION CRITERIA	PS	PS	PS	PS	PS	U	PS	PS	PS	PS

Weekly Cycle

The practice of Weekly Cycle (WC) has seen the introduction of three additional elements to that of the original practice. Each of the additional elements takes inspiration from the practice of the Daily Scrum Meeting (DSM) from the Scrum methodology. As was identified in the literature review the purpose of the DSM is to provide the team with an update as relating to the current situation with the project by answering three questions (Sutherland, (Sutherland, 2005):

- What tasks have been completed in the previous 24 hours?
- What obstacles were found when completing these tasks?
- What tasks are expected to be completed in the next 24 hours?

These questions therefore provide effective guidelines for potential improvement with the WC practice. As the primary task of this practice is to produce a piece of working software by the end of each week it would be further productive to set aside time for a short review meeting, to discuss what has been completed, how the work completed complies with the time constraints put in place and also allowing for re-evaluation of project resources, thus increasing efficiency. As Law and Charron (2005) state, issues raised within the meeting could be solved quickly, as a team, therefore providing a variety of solution paths and reducing the potential existence of nine identified project failures.

WEEKLY CYCLE	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Produce a piece of working software each week								•		
Develop according to priority of tasks	•		•			•		•		•
Persistent problems identified		•			•					
Progress of development period measurable	•		•	•	•			•	•	•
Re-evaluate resource management					•					•
SATISFACTION CRITERIA	PS	PS	PS	PS	PS	PS	U	PS	PS	PS

Continuous Integration

Continuous Integration (CI) witnesses the introduction of one new element. By using this practice as a tracking tool rather than just an integration tool it enables the possibility to increase the support given to continually controlling and managing highlighted risks (PF2) throughout the project life.

CONTINOUS INTEGRATION	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Faults identified immediately									•	
Progress measurable	•		•			•		•		•
Method for tacking faults		•								
SATISFACTION CRITERIA	PS	PS	PS	U	U	PS	U	PS	PS	PS

Scrum Master

As was highlighted in the literature review the role of the Scrum Master (SM) is to oversee that the principles of Scrum are upheld and that the correct structure to meetings is followed throughout the development process. It is at the meeting that the introduction of

the element is to be implemented. At present, the SM is responsible for ensuring that the three questions detailed in the literature review are answered by each individual from the team. Whilst this is very effective, the discussion focuses on what has been done and what will be completed based on the information that the development team have. As has already been stated the Product Backlog is updated as soon as something within the project changes but to ensure that the development team are fully aware of any changes it would be an effective task for the SM to vocalise any updates to ensure that the development team are kept as up to date on project information as possible.

SCRUM MASTER	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Upholds scrum principles										
Controls scrum meeting		•						•	•	•
Acts as team liaison						•	•			•
Ensures any updates are vocalised		•		•		•	•			
SATISFACTION CRITERIA	U	PS	U	PS	U	PS	PS	PS	PS	PS

Product Backlog

The addition of the new element that dictates that time estimations should also be recorded within the Product Backlog (PB) increases the possible support which this practice can offer in relation to the identified project failures. As the table below illustrates the new element brings a source to overcoming PF6 and PF10 which were originally unsupported.

The information which is entered into the PB must be as a result of discussion between the client and the development team, therefore if it has not been discussed then it cannot enter the PB and this increasing the opportunity for errors to materialise throughout development. PF6 states that clients were not involved enough in the making of schedules for the project. By instigating that time estimations must exist in the PB it ensures that the created schedules appeal to both developers and to the client.

Resolution to PF6 then enables PF10 to become better supported. The failure of the development team under delivering on their initial commitments can be better supported and overcome as the discussion of time estimates will provide both stakeholders the understanding as to the duration of a task thus enabling the setting of realistic goals which are manageable.

PRODUCT BACKLOG	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Every requirement/ piece of functionality recorded	•			•			•	•	•	
Made available to all stakeholders							•			
Kept up to date	•	•		•						
Time estimations recorded						•		•		•
SATISFACTION CRITERIA	PS	PS	U	PS	U	PS	PS	PS	PS	PS

Sprint Plan Meeting

The first new element within the Sprint Plan Meeting (SPM) is an extension of a key concept from the Daily Scrum Meeting (DSM), that is, the awareness of potential forthcoming risks. As the DSM dictates that development team members should bring awareness to others as to the risks which lay ahead in the next twenty-four hours, the new element within the SPM adopts the same ideas but extends it to risk which lay ahead in the thirty-day sprint. This achieves the provision of backing given to PF2 minimising the possible existence of risks in the project development.

The second element to be introduced in this practice commands that rather than set a budget for the project as a whole, a budget should be set per sprint. As modern software development projects continually evolve, it is difficult to declare a budget for a project which may change several times before its completion, thus often resulting in the project finishing over budget (PF9). By setting a budget on a Sprint basis it will provide a greater control of finances avoiding unnecessary costs.

SPRINT PLAN MEETING	PF1	PF2	PF3	PF4	PF5	PF6	PF7	PF8	PF9	PF10
Client opportunity to provide input				•		•	•			
Prioritise requirements to be developed	•							•		•
Delegate tasks			•							
Made aware of risks of forthcoming sprint		•								
Set a budget for the sprint									•	
SATISFACTION CRITERIA	PS	PS	PS	PS	U	PS	PS	PS	PS	PS

APPENDIX D – Dimension 2 Raw Data

Phase Analysis

XP	FY	SD	LS	LG	RS	CN	TG	Total
PRE-DEVELOPMENT	1	1	0	1	1	1	1	6
DEVELOPMENT	1	1	1	1	1	1	1	7
POST-DEVELOPMENT	0	1	0	1	0	1	1	4
TOTAL	2	3	1	3	2	3	3	**17**
DOA	67%	100%	33%	100%	67%	100%	100%	**81%**

SCRUM	FY	SD	LS	LG	RS	CN	TG	Total
PRE-DEVELOPMENT	1	1	0	1	1	1	1	6
DEVELOPMENT	0	1	0	1	1	1	1	5
POST-DEVELOPMENT	0	1	0	1	0	1	1	4
TOTAL	1	3	0	3	2	3	3	**15**
DOA	33%	100%	0%	100%	67%	100%	100%	**71%**

Revised Phase Analysis

XP REVISED	FY	SD	LS	LG	RS	CN	TG	Total
PRE-DEVELOPMENT	1	1	0	1	1	1	1	6
DEVELOPMENT	1	1	1	1	1	1	1	7
POST-DEVELOPMENT	0	1	0	1	1	1	1	5
TOTAL	2	3	1	3	3	3	3	**18**
DOA	67%	100%	33%	100%	100%	100%	100%	**86%**

SCRUM REVISED	FY	SD	LS	LG	RS	CN	TG	Total
PRE-DEVELOPMENT	1	1	0	1	1	1	1	6
DEVELOPMENT	0	1	0	1	1	1	1	5
POST-DEVELOPMENT	0	1	0	1	1	1	1	5
TOTAL	1	3	0	3	3	3	3	**16**
DOA	33%	100%	0%	100%	100%	100%	100%	**76%**

Practice Analysis

XP	FY	SD	LS	LG	RS	CN	TG	Total
Sit Together	1	1	0	1	1	1	0	5
Whole Team	1	0	0	1	0	1	0	3
Informative Workplace	1	1	0	1	1	1	1	6
Energised Work	0	1	1	0	0	0	0	2
Pair Programming	1	1	0	1	1	1	1	6
Stories	1	1	1	1	1	1	0	6
Weekly Cycle	1	1	0	0	1	1	1	5
Quarterly Cycle	1	0	0	1	0	1	1	4
Slack	1	0	0	0	0	0	0	1
Ten Minute Build	0	1	1	0	1	0	1	4
Continuous Integration	1	1	1	1	1	0	1	6
Test First Programming	0	0	1	0	1	0	1	3
TOTAL	9	8	5	7	8	7	7	**51**
DOA	75%	67%	42%	58%	67%	58%	58%	**61%**

SCRUM	FY	SD	LS	LG	RS	CN	TG	Total
Scrum Master	1	1	0	1	1	1	1	6
Product Backlog	1	1	0	0	1	1	1	5
Scrum Team	1	1	0	0	0	1	0	3
Daily Scrum Meeting	1	1	0	1	1	1	1	6
Sprint	1	1	0	1	1	1	0	5
Sprint Review Meeting	1	1	0	1	0	1	1	5
Sprint Plan Meeting	1	1	0	1	1	1	0	5
TOTAL	7	7	0	5	5	7	4	**35**
DOA	100%	100%	0%	71%	71%	100%	57%	**71%**

Revised Practice Analysis

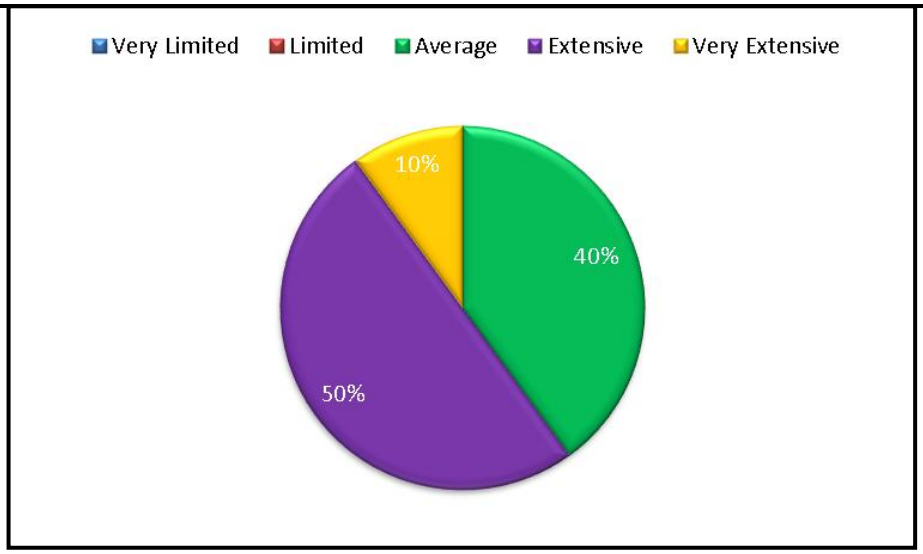
XP REVISED	FY	SD	LS	LG	RS	CN	TG	Total
Sit Together	1	1	0	1	1	1	1	6
Whole Team	1	0	0	1	0	1	0	3
Informative Workplace	1	1	0	1	1	1	1	6
Energised Work	0	1	1	0	0	0	0	2
Pair Programming	1	1	0	1	1	1	1	6
Stories	1	1	1	1	1	1	0	6
Weekly Cycle	1	1	0	0	1	1	1	5
Quarterly Cycle	1	0	0	1	0	1	1	4
Slack	1	0	0	0	0	0	0	1
Ten Minute Build	0	1	1	0	1	0	1	4
Continuous Integration	1	1	1	1	1	1	1	7
Test First Programming	0	0	1	0	1	0	1	3
TOTAL	9	8	5	7	8	8	8	**53**
DOA	75%	67%	42%	58%	67%	67%	67%	**63%**

SCRUM REVISED	FY	SD	LS	LG	RS	CN	TG	Total
Scrum Master	1	1	0	1	1	1	1	6
Product Backlog	1	1	0	1	1	1	1	6
Scrum Team	1	1	0	0	0	1	0	3
Daily Scrum Meeting	1	1	0	1	1	1	1	6
Sprint	1	1	0	1	1	1	1	6
Sprint Review Meeting	1	1	0	1	1	1	1	6
Sprint Plan Meeting	1	1	0	1	1	1	1	6
TOTAL	7	7	0	6	6	7	6	**39**
DOA	100%	100%	0%	86%	86%	100%	86%	**80%**

APPENDIX E – Questionnaire Results

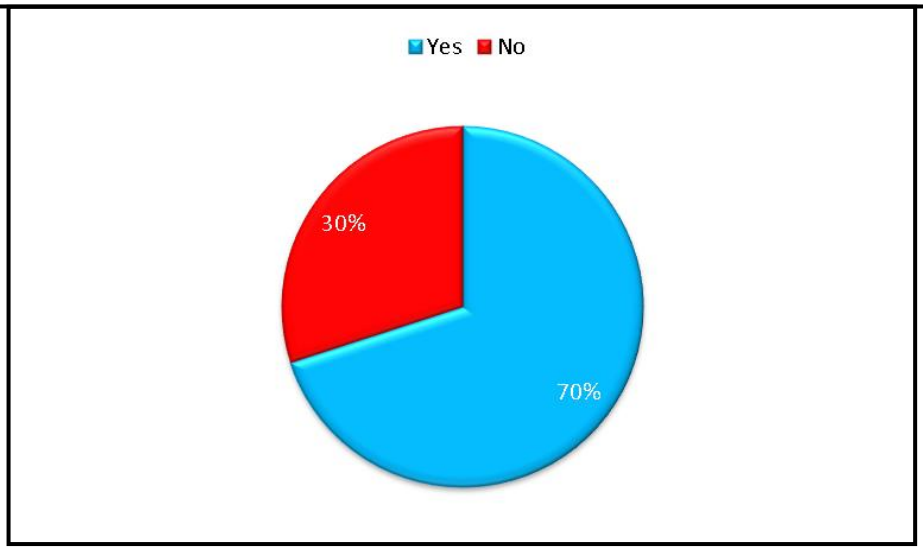
1. How would you rate your knowledge of Methodologies?

- Very Limited
- Limited
- Average
- Extensive
- Very Extensive



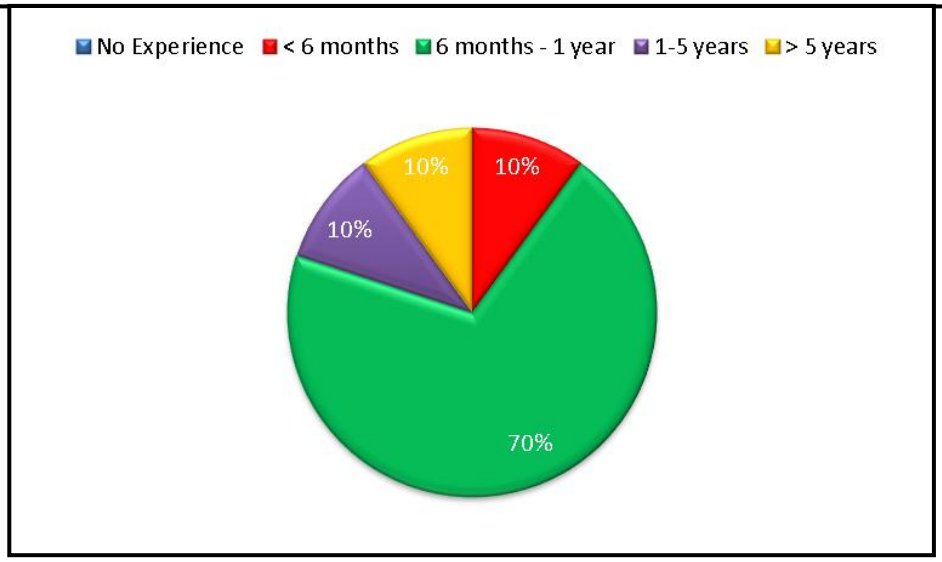
2. Does your current organisation adopt the use of Agile Methodologies?

- Yes
- No



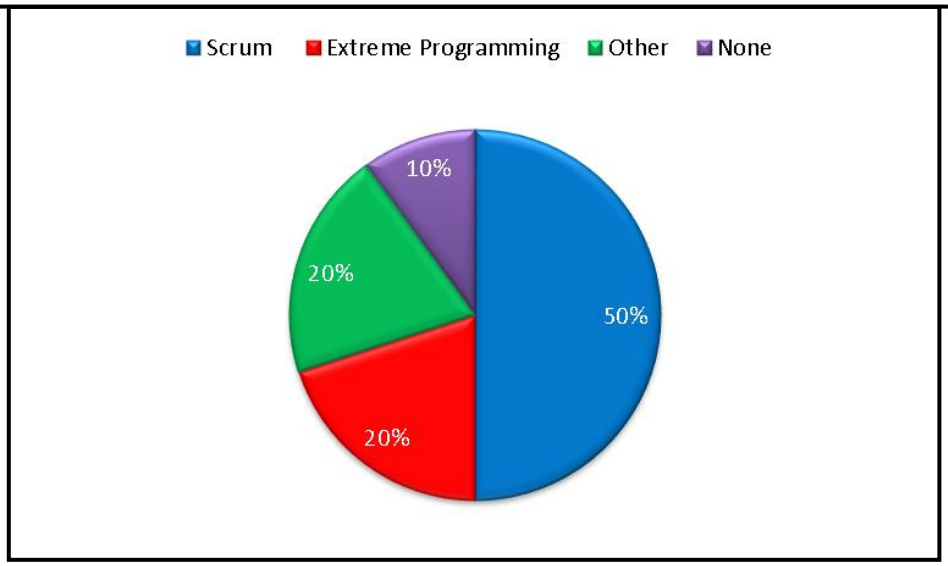
3. How would you describe the level of experience towards Agile Methodologies of project members within a typical software development project?

- No Experience
- Less than 6 months
- 6 months – 1 year
- 1 – 5 years
- More than 5 years



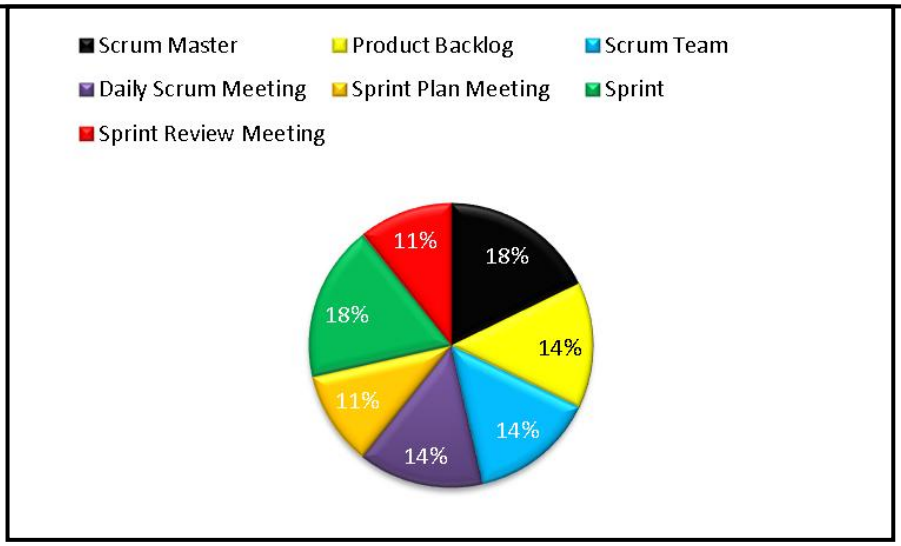
4. Which Agile methodology does your organisation implement? (Select the most popular)

- Scrum
- Extreme Programming
- None
- Other



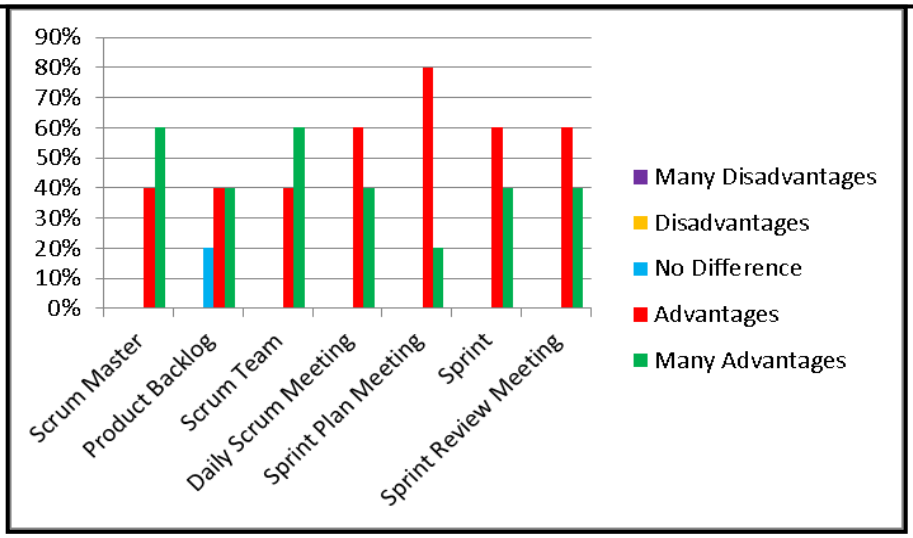
5. Which practices of Scrum does your organisation implement? (Select all that apply)

- Scrum Master
- Product Backlog
- Scrum Team
- Daily Scrum Meeting
- Sprint Plan Meeting
- Sprint



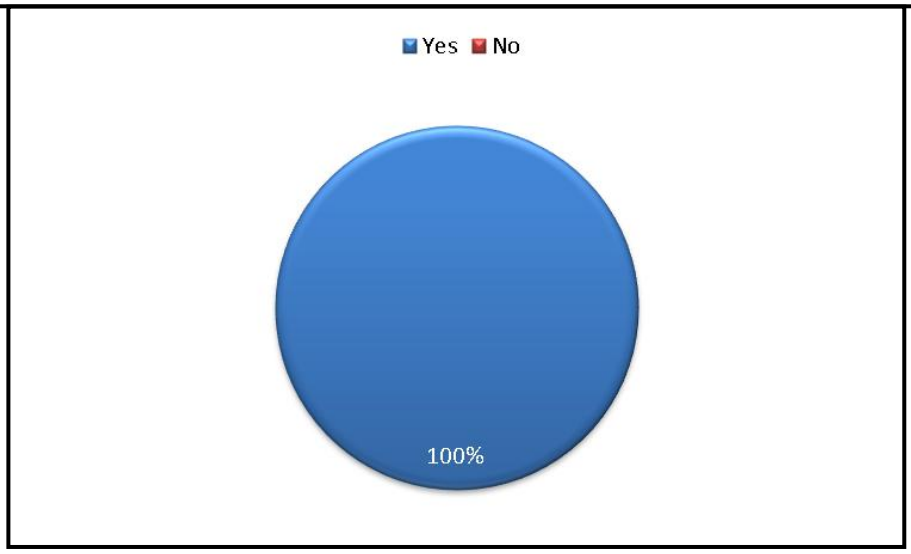
6. How would you rate the influence that each practice introduces to a software development project?

	Many Disadvantages	Disadvantages	No Difference	Advantages	Many Advantages
Scrum Master					
Scrum Team					
Product Backlog					
Daily Scrum Meeting					
Sprint Plan Meeting					
Sprint					
Sprint Review Meeting					



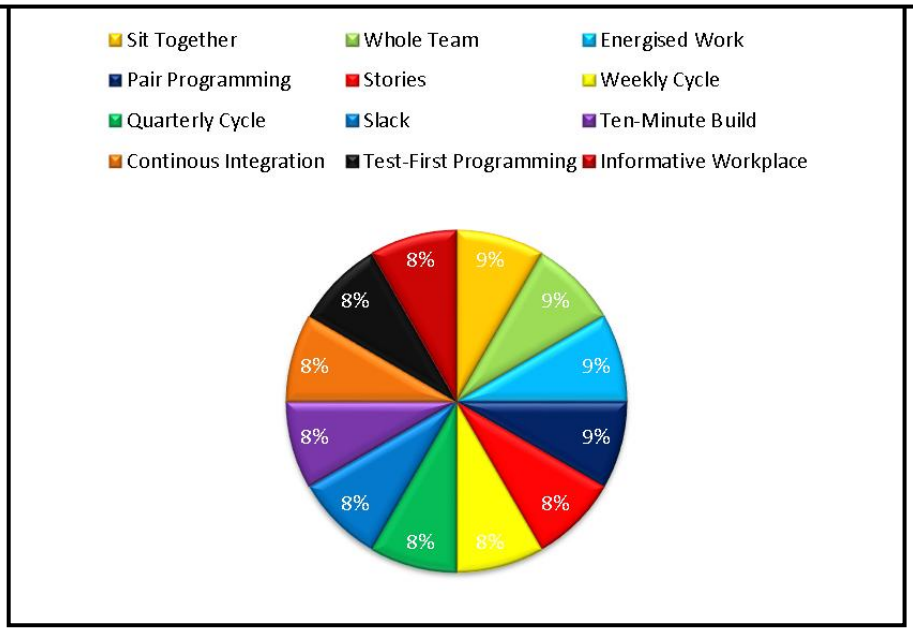
7. Having implemented Scrum would you be happy to use it again in future projects?

- Yes
- No



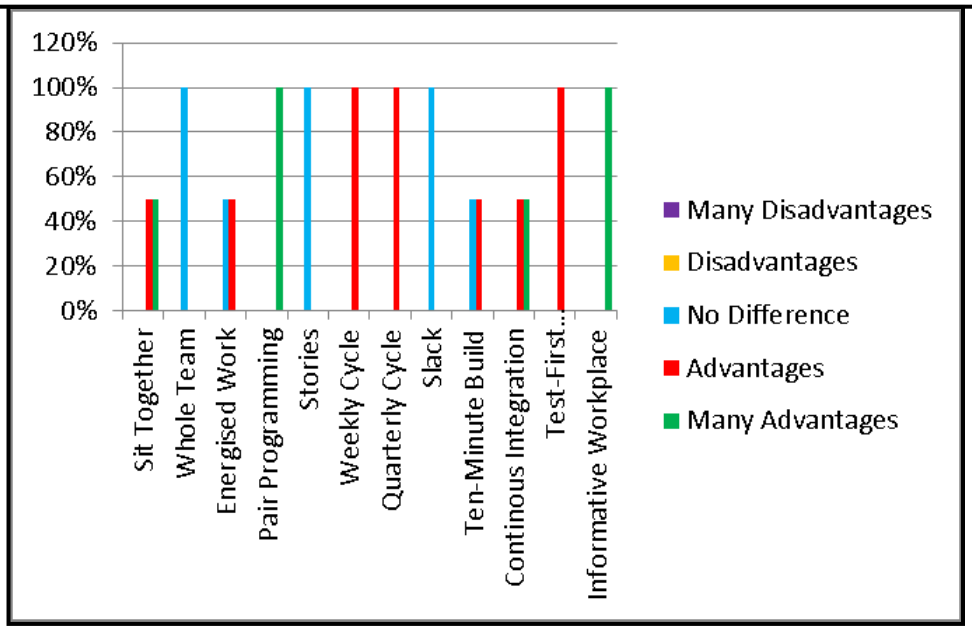
8. Which practices of Extreme Programming does your organisation implement? (Select all that apply)

- Sit Together
- Whole Team
- Energised Work
- Pair Programming
- Stories
- Weekly Cycle
- Quarterly Cycle
- Slack
- Ten-Minute Build
- Continuous Integration
- Test-First Programming
- Informative Workplace



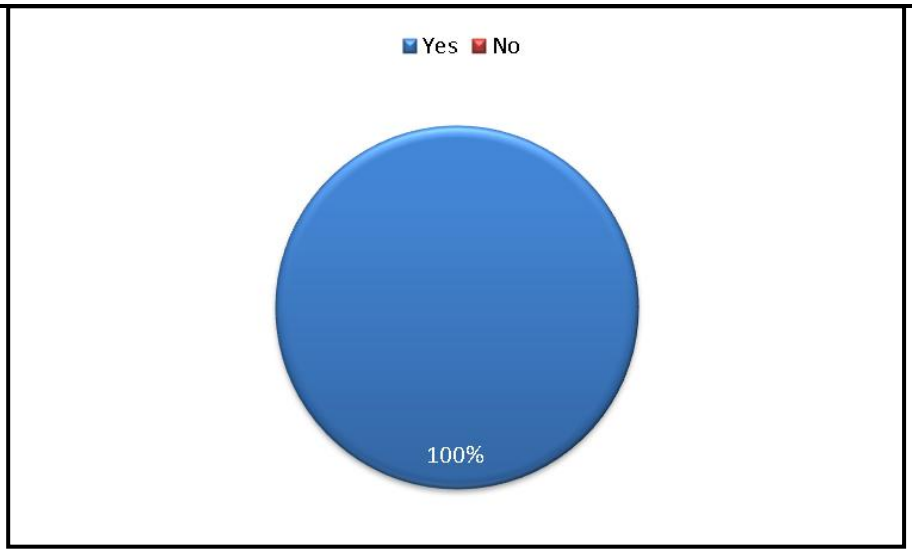
9. How would you rate the influence that each practice introduces to a software development project?

	Many Disadvantages	Disadvantages	No Difference	Advantages	Many Advantages
Sit Together					
Whole Team					
Energised Work					
Pair Programming					
Stories					
Weekly Cycle					
Quarterly Cycle					
Slack					
Ten-Minute Build					
Continuous Integration					
Test-First Programming					
Informative Workplace					



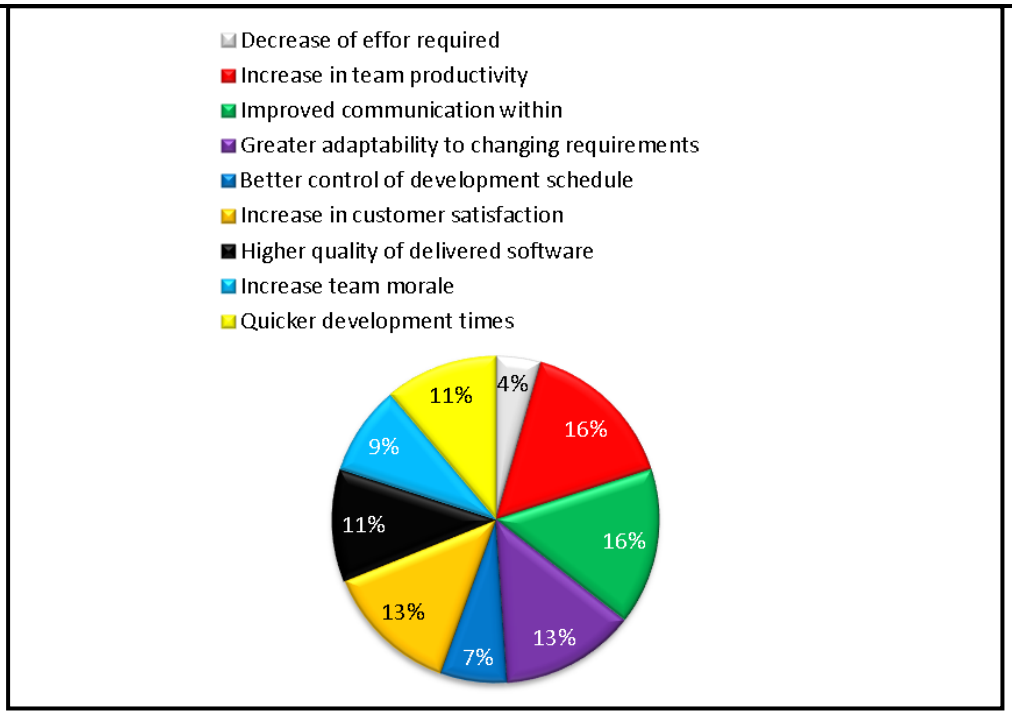
10. Having implemented Extreme Programming would you be happy to use it again in future projects?

- Yes
- No



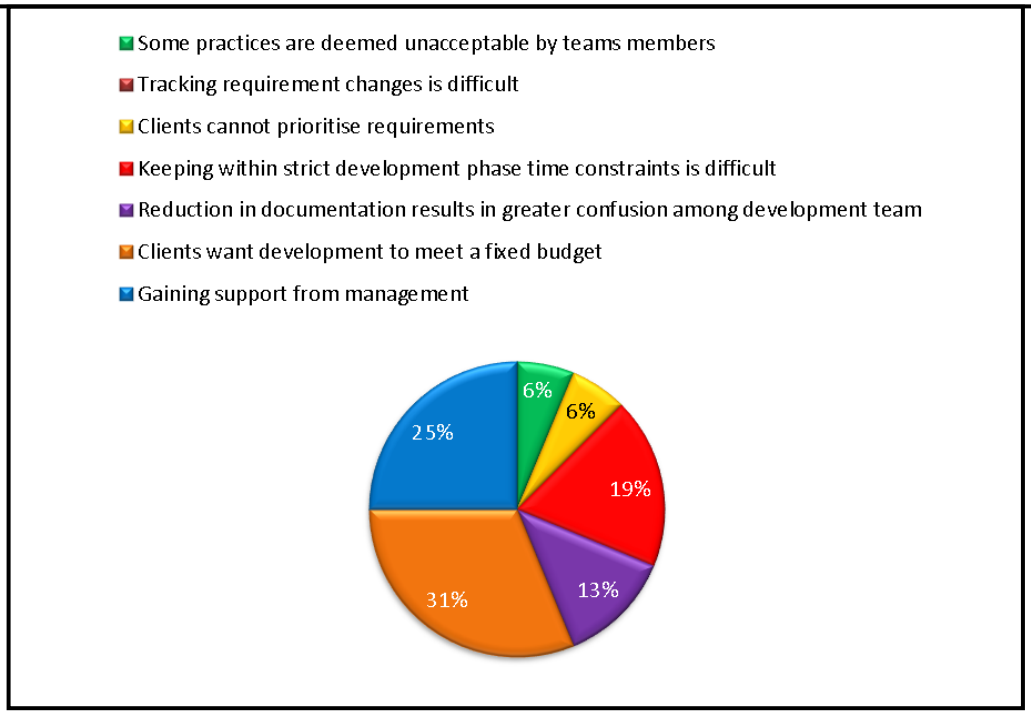
11. In your opinion what advantages has using an agile methodology brought? (Select all that apply)

- Decrease of effort required
- Improved communication within
- Better control of development schedule
- Decrease of effort required
- Increase in team productivity
- Greater adaptability to changing requirements
- Increase in customer satisfaction
- Higher quality of delivered software



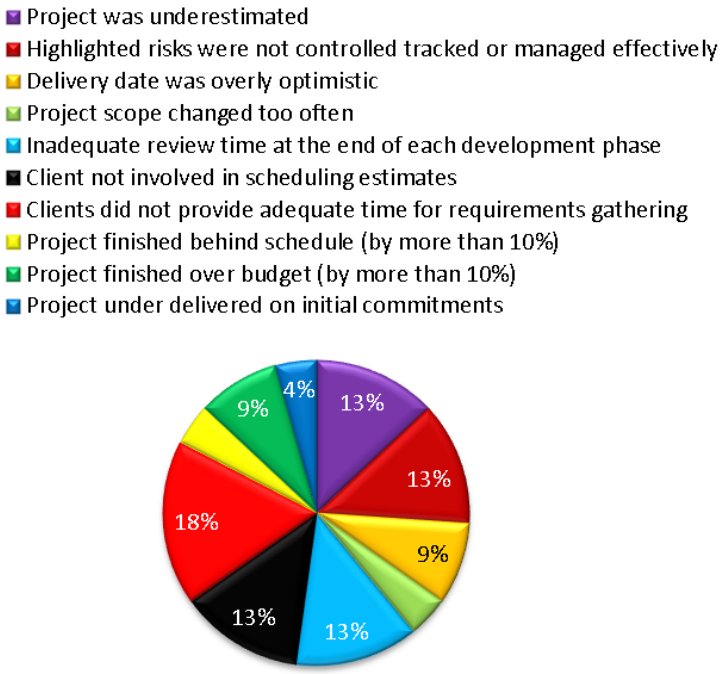
12. In your opinion what disadvantages has using an agile methodology brought? (Select all that apply)

- Some practices are deemed unacceptable by team members
- Tracking requirement changes is difficult
- Clients cannot prioritise requirements
- Keeping within strict development phase time constraints is difficult
- Reduction in documentation results in greater confusion among development team
- Clients want development to meet a fixed budget
- Gaining support from management



13. On projects which you have been included, what would you identify as being the biggest problem areas? (Select all that apply)

- Project was underestimated
- Highlighted risks were not controlled, tracked or managed effectively
- Delivery date was overly optimistic
- Project scope changed too often
- Inadequate review time at the end of each development phase
- Client not involved in scheduling estimates
- Clients did not provide adequate time for requirements gathering
- Project finished behind schedule (by more than 10%)
- Project finished over budget (by more than 10%)
- Project under-delivered on initial requirements



14. Are there any additional comments that you would like to add relating to your experience of using Agile?

- It can be difficult to get developers to start using agile methodology when they are so used to traditional project management methodologies, such as Waterfall.
- Some customers are not yet used to intense interaction while preparing, executing and delivering a project despite all the benefits. Time will resolve this, though.
- The agile methodology allows project teams to maximize waste. That is to say we are able to deliver the highest business valued features first and drop the lowest features once the project nears completion becaise the business recognise that these features are now worthless. Problems using agile with senior managers who are absolutely obsessed with 'on time' and 'on budget' approach. What's wrong with delivering value all the time and control the budget to continue to deliver value when change inevitably comes along?
- Agile well understood and applied increases: happiness, productivity, focus, quality, transparency collaboration, trust, success (...)