Final Honours Report

"Investigation into how the quality practices of Extreme Programming can be adapted and integrated with the quality process areas of the Capability Maturity Model for Development to improve the software development process."

Professor Calculus
2004xxxxx
Computing (Software Engineering, BHCG4_1) Year 4

Project Supervisor: **Dr. Richard Foley**
Second Marker: **Dr. Edwin Gray**

Submitted for the Degree of BSc in Computing 2007 - 2008

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

Signed:_____ **Date_____**

Abstract

The desire to improve how the industry develops software is critical in order to address software disasters that are prominent today. The UK Government spends £14 billion per year on public sector IT with only 30% of government IT projects being deemed successful and reports have identified that nearly two thirds of all United State IT projects fail. The emergence of lightweight Agile methodologies, due to their reputation to deliver high-quality, functioning software at a fraction of the time and cost of traditional methods, has sparked a debate in the IT world. By focusing on software quality, this project investigates XP practices and how they relate to the quality aspects of the CMMI for Development Version 1.2, an established quality framework. This will provide the software development industry with further research into understanding relationships between the quality process areas of the CMMI-Dev and XP quality practices that could improve the manner in which current organisations develop software with quality control. A review was conducted into the generic activities of software development that revealed common development activities that are standard in the majority of software development projects. An investigation was then conducted into Extreme Programming's quality practices and their relation to the general software development activities. The characteristics and quality process areas of the CMMI-Dev were then analysed in order to provide an understanding into how they related to a specific generic development activity. By analysing the CMMI-Dev process areas and the XP practices in relation to common software development activities, a framework was developed that showed each process area of the CMMI-Dev and XP practice grouped under common software development activities that allowed an explanatory experiment to be conducted to identify relationships between CMMI-Dev and XP. This study has revealed that when XP is mapped to the CMMI-Dev under generic activities of software development, the results show that there are areas where XP and the CMMI-Dev do not map such as design, implementation, and SQA areas. The low mapping between project management activities in XP and the CMMI-Dev, highlights the lack of project management in XP and supports the objection to using XP in large organisations because it barely touches the management and organisational issues that the CMMI emphasises. However, the continuing argument that suggests that XP cannot be used in the rigorous, process intensive CMMI is not supported by this study. XP uses well defined techniques and practices and has shown in this study that it highly satisfies the CMMI-Dev in the verification and validation activities by emphasising testing. The CMMI-Dev informs organisations what to do where as XP is a set of best practices that has information on how to develop software, thus by using a combination of both, developing software in the future may be more successful. This study can thus establish that the traditional quality framework of CMMI-Dev and agile best practices of XP can complement one another in areas of software development.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor, Dr. Richard Foley for his advice, support and encouragement throughout the project. I would also like to thank my family for their support and understanding during the course of this year. This study could not have been completed without your support and help.

1.0 Introduction

1.1 Background

Software Engineering Today

In the constantly evolving and fast paced software development industry of today, the overall goal is to quickly produce software that is of high quality. A recent report produced by the leaders of IT project and value performance, The Standish Group (Rubinstein 2007), have identified that nearly two thirds of all United State IT projects fail. Findings show that projects fail to meet set standards with 19% being outright failures and 46% either overrunning their budget or not being delivered on time. Joe Harley, the chief information officer of the Department for Work and Pensions stated that the UK Government spends £14 billion per year on public sector IT in the UK (Tony Collins 2007). Whilst addressing such problems at the Government UK IT Summit in May 2007, Mr. Harley also reported that only 30% of government IT projects are actually successful. The current high failure rate of IT projects within the UK Government and United States highlights a major problem in the development of software worldwide. There are large amounts of capital being invested in software development, yet there are still too many project failures. To address these worldwide software development issues, organisations must realise that there are improved ways to develop software.

The Problem

Hinde (2005) states the reasons for project failures a decade ago seem to be the recurring factor in the failures that the software developments are experiencing today, the overall lack of good software engineering practices and quality processes. With traditional development methods, the process has a strict, lengthy analysis and documentation of requirements at the beginning of every project with the client expected to define 100% of their requirements at the beginning. This means change throughout the rest of the project is restricted and is the leading factor in why projects are delivered late, do not meet the client's requirements or become over budget (Crosman 2007).

Agile Software Development

The two most popular forms of developing software are either described as Agile lightweight processes or traditional heavyweight processes with no shared ground in-between. As the industry has evolved, Software Quality Management systems have been developed to accompany the appropriate process methodology. Process methodologies that organisations have been using to develop their high quality software have also evolved over time. The emergence of lightweight Agile methodologies, due to their reputation to deliver high-quality, functioning software at a fraction of the time and cost of traditional methods, has sparked a debate in the IT world (Hinkle 2007).

Extreme Programming (XP) is an Agile software engineering methodology that has set practices that encourage particular extreme values. XP is a disciplined approach to software development and its reputation has grown worldwide but yet to become mainstream (Cane 2007). The Agile Alliance (Kent Beck et al. 2001), the founders of Agile practices, believe that by exercising these set practices, this will lead to a development process that is more responsive to client needs than traditional methods whilst creating software of a higher quality.

Agile methods (Abrahamsson et al. 2003, Damian 2002a) are now being used to address the reiterating key problems in software development; that software takes too long to develop, costs too much, and does not operate fully when eventually delivered. McCauley (2001) believes that XP emphasises speed and simplicity and seeks to avoid prescribing cumbersome and time-consuming processes that add little value to the software product. Instead, the focus is on individuals and interactions, working software, client collaboration, and fast response to changes which are included in the Agile Manifesto (Kent Beck et al. 2001). XP is an attempt to satisfy the industry for a more lightweight and faster development process. Large organisations often feel that agile methods are not applicable to themselves and that they are easier to implement in small/medium enterprises. This results in why large organisations are cautious of them and prefer to adopt the tried and tested CMMI discusses Paulk (2001).

Traditional Software Development

Wall Street organisations in the past have been focusing on software outsourcing and project cost reduction but now recently have turned their attention to efficient software development (Crosman 2007). Organisations are now investing resources in traditional software development processes like the Capability Maturity Model Integration (CMMI) that enforces disciplined, processes and measurements around software development.

The CMMI is a five level model that attempts to quantify a software organisation's capability to consistently and predictably produce high quality software products. The creators of the CMMI, Carnegie Mellon University's Software Engineering Institute (SEI) (Carnegie Mellon University 2008), describe the CMMI as a process improvement approach that provides organisations with the essential elements of effective processes. Paulk et al (1995) explains each development stage or maturity level distinguishes an organisation's software process capability and provides a point of reference for appraising current processes. The SEI published in a report (Gibson, Goldenson & Kost 2006) that sixty organisations measured increases of performance in the categories of cost, schedule, productivity, quality and client satisfaction whilst using the CMMI. Although the CMMI does produce successful results, it is not accepted by all in the software development industry as it is viewed as a heavyweight and over bureaucratic means of developing software (Dickerson 2001).

Although there are many benefits to implementing the CMMI, it cannot be applied to all projects thus is criticised in reports (Bollinger 1991, F. G. Wilkie, D. McFall & F. McCaffery 2005) as showing omissions and weaknesses. In the study by Staples et al (2007), forty software development organisations are surveyed on their reasons for not

implementing the CMMI. The results of the study show that the organisations either found their selves to be too small an organisation, it being too costly to introduce, having no time to implement or they were already using another software process improvement program that suited their type of development. This enforces the idea that the CMMI does not suit all organisations / projects but yet is one of the leading processes used in the world today to produce software. Agile software developers insist that Agile software development is the best route to speed and quality of software development compared to traditional approaches like the CMMI (Lindvall et al. 2004) thus it would be useful to investigate the quality practices of both.

The Solution

Baskerville et al (2006) discusses the benefits of Agile methods over traditional approaches which include an early return on investment giving the opportunity to show early results to the client. Agile methods are very responsive to changing requirements with processes, tools, roles and principles that allow a team and an organisation to embrace change rather than reject, control or suffer from change which studies (Willoughby 2005) have shown the CMMI often enforces. The CMMI identifies what organisations should do to improve quality practices but does not explicitly state how the organisation can improve, where as Agile methods are a set of best practices that contain specific information on how to implement software.

By applying the quality practices of Extreme Programming to the CMMI, this study will provide research into an improved manner of developing software with quality control. A senior member of the Software Engineering Institute, Paulk (2001), argues that quality frameworks like the CMMI can complement Agile practices to overcome the issues that arise when using traditional methods with a Quality Management System. Whilst there is obvious evidence that the CMMI can improve an organisation's quality process, Turner & Jain (2002) argue that Agile methods can also improve a project whilst emphasising both approaches have much in common. Their study shows that both have flaws in developing software, but that there are phases in a project that could benefit from aspects of each development process.

Some companies in the software industry have traditionally outsourced software development abroad (Crosman 2007) and may be interested in this study as it will reveal insight into Agile methods blended with their current traditional Quality Management Systems. For organisations who have Quality Management Systems securely in place with a traditional process but are interested in Agile development, this may give an insight into which division of their Quality Management Systems could be modified to accept Agile practices.

This project focuses on XP practices and how they relate to quality aspects of the CMMI for Development Version 1.2 (CMMI-Dev), the most recent version of the CMMI. This provides the software development industry with research based incite into understanding relationships between the quality practices of the CMMI-Dev and XP which may improve current or future quality frameworks.

1.2 Project Outline & Research Question

The outline of the project involves an explanatory experiment that identifies relationships between the CMMI-Dev quality process areas and the quality practices of XP.

Baker (2006) discusses that organisations with Agile software practices often receive timely, cost-effective solutions of sufficient quality whilst organisations that use traditional quality processes often benefit from industry-recognised certifications and robust process improvement mechanisms. The point being stressed is that rarely does a single, large IT organisation succeed to achieve both sets of benefits. This study highlights quality practices used in both XP and the CMMI-Dev that share quality aspects and discusses the relations between each which may allow for future work to produce a quality improvement process or framework that is a hybrid of both.

Research Question

The research question was developed in order to focus the experiment and achieve the project aim. The research question is:

"How successfully can the quality practices of Extreme Programming be adapted and integrated with the quality process areas of the Capability Maturity Model for Development?"

In the following section, the study identifies objectives that were used to achieve the project aim. The objectives gave direction to the project and focused on areas that were investigated during the experiment.

Aims & Objectives

The purpose of this study was to identify how successfully the quality practices of XP can be incorporated into the CMMI-Dev framework. The results obtained from the experiment provide software development organisations with a clear and precise understanding of how XP practices could be integrated with the CMMI-Dev.

In order for the project to succeed, the study addressed several objectives.

1. Investigate the generic activities of a standard software development process.

Objective Purpose
This revealed the standard development activities that are common in the majority of development processes. This information on generic activities was useful as it provided a common framework that was used in the experiment to show what practises of XP and process areas of the CMMI-Dev related under the same generic development activity.

2. Review Extreme Programming's quality practices and relate to general software development activities.

Objective Purpose
This revealed details on XP through a literature review and laid a foundation to fully understand its quality practices and affects on the development process. A further understanding was developed to reveal more information about the software engineering practices and quality processes involved in XP, which allowed each practice to be grouped under a general development activity.

3. Review the characteristics and quality process areas of the CMMI-Dev and relate them to standard software development activities.

Objective Purpose
This uncovered detailed information through a literature review which provided an understanding into the process areas of the CMMI-Dev. The process areas were related to a generic development activity to provide a high level grouping of what standard software development activity the process area fulfils.

4. Identify the CMMI-Dev process areas and the XP practices that can be included under the generic framework for software development activities.

Objective Practice
The overall objective of the literature review was to identify what each practice and process area does and to identify where each could be grouped under the common framework of software development activities. This allowed the practices and process areas that could not be grouped under the common framework to be eliminated from the study before proceeding to the experiment.

5. Incorporate XP quality practices into the CMMI-Dev process areas under a generic development activity.

Objective Purpose
This exposed relationships between the quality aspects of XP and the CMMI-Dev through the explanatory experiment. Each process area and practice was enumerated under a generic development activity to reveal relationships between them.

Hypothesis

By performing research into the specific area of combining the quality practices of XP with the CMMI-Dev, a hypothesis was developed to identify what might have been discovered by the study. Justification has also been provided by referring to literature that reveals there is a need to investigate this area of software process improvement.

- Only certain practices of XP can be introduced to the disciplined, traditional Quality Management System of the CMMI-Dev.

Justification

A recent project by Intel Shannon (Fitzgerald, Hartnett & Conboy 2006), part of Intel's Infrastructure Processor Division in West Ireland, has shown benefits to combining Agile methods into their current quality framework. The main advantage being the delivery of the software within project schedule although there were issues with quality within the code. The report also shows the aspects of Agile methods that cannot be mixed within their quality framework.

1.3 Report Structure / Contents

Literature Review

The purpose of the literature review, chapter two, was to review and discuss XP and the CMMI-Dev in relation to quality to allow greater understanding in the field. By analysing the XP practices and CMMI-Dev process areas to generic software development activities, a framework was established that provided a foundation on which the experiment could be conducted. By grouping each practice and process area into a similar generic software engineering activity, it allowed the experiment to reveal relationships between the CMMI-Dev and XP.

Methodology

The purpose of the methodology, chapter three, was to present in detail the specifics of the primary research approach. Included is thorough discussion and critical evaluation of literature to justify why the selected research approach was selected for the experiment. The project involved an explanatory experiment that analysed the CMMI-Dev process area specific goals, and XP's practices to identify relationships between them. The type of primary research method used was an explanatory experiment as it enabled the study to highlight or disregard links between factors by means of theory.

Evaluation

The evaluation, chapter four, carries out an explanatory study that considers only the CMMI-Dev process areas and XP practices identified in the literature review as being relevant to the study. For each process area, a mapping between its specific practices to fulfill each goal and the XP practice was carried out. After that, a coverage rating for each practice was established and a coverage percentage of each process area was calculated to determine how applicable the XP practice(s) were during the experiment. The results were then grouped and a complete view of the CMMI process areas coverage by XP practices was generated. Each mapping and the results are discussed in the evaluation chapter under each software development activity.

Final Discussions & Conclusions

The final discussions and conclusions section of the report, chapter five, presents the final overall conclusions that can be drawn from the results of the evaluation method detailed in chapter four of the project. The results are discussed within the generic development activities of a project and include individual conclusions and reference to how they link to the work of others with the chapter concluding with what future work could be undertaken following the results of this project.

2.0 Literature Review

The purpose of the literature review was to complete the following specific objectives and questions:

- Review and discuss Extreme Programming (XP) and its practices in relation to quality.
- Do the Extreme Programming practices relate to standard software development activities?
- Review and discuss the Capability Maturity Model Integration for Development V1.2 (CMMI-Dev), the process areas and relate them to quality.
- Do the process areas relate to generic software development activities?

By answering and completing the objectives and questions above, the literature review was able to reveal a high level relation between the practices and process areas with standard development activities. The explanatory experiment was then given a foundation on which to build upon as each practice and process area was grouped into a similar generic software engineering activity, revealing relationships between each in the experiment.

2.1 The Generic Software Development Lifecycle

The development activities that were used are taken from Pressman and Sommerville, renowned for their commitments to software development. Pressman (2000) includes in his definition of generic development practices, Software Project Management, Software Quality Assurance, Software Configuration Management and Risk Management. Whilst Sommerville (2006) believes Software Requirements & Specification, Design, Implementation and Validation & Verification activities should be used in a standard development lifecycle. On researching each practice and process area from XP and the CMMI-Dev and relating to generic activities, the experiment identified where each process area of CMMI-Dev and practice of XP related. This may be useful to an organisation that is looking to improve on certain aspects of their lifecycle by introducing Agile practices or similarly adding CMMI-Dev process areas. Below are brief summaries of what each generic activity involves.

Software Requirements & Specification

The aim of this section is to define the functionality of the software, stakeholder needs and constraints of the system. A Software Requirements Specification (SRS) is a complete description of the behavior of the system to be developed. Activities that are included in this section are requirements elicitation, requirement analysis and requirement specification.

Design

This section produces a design plan of a solution that must meet the requirements that have been set by the development team and the client for the project to be successful (Sommerville 2006). Different designs are data designs, architectural designs, interface designs and component level designs.

Implementation

Implementation is the execution of the design and requirements, the physical coding of the product.

Validation & Verification (V&V)

Boehm (1984) concisely expresses the difference between validation and verification in the following questions.

- 'Validation: Are we building the right product?'
- 'Verification: Are we building the product right?'

The role of verification involves checking that the software conforms to its specification whilst validation checks that the product meets the client's needs by operating in the expected manner. Activities can take the form of formal, functional, testing, runtime and software validation and verification.

Software Project Management

Good management cannot guarantee project success but bad management usually results in project failure states Sommerville (2006). Software management involves planning and scheduling whilst satisfying standards and monitoring progress to check that development is on time and within budget. Management activities include project planning and scheduling, feasibility studies, monitoring and reviewing, proposal writing and being able to perform and understand risk analysis.

Software Quality Assurance (SQA)

Software Quality Assurance consists of monitoring the software engineering processes and methods used to ensure quality. It does this by means of audits of the quality management system under which the software system was created. Pressman (2000) emphasises three important points to define software quality. They are software should conform to requirements, specified standards should be defined to guide the manner in which software is engineered, and if software fails to meet implicit requirements (ease of use, good maintainability) then software quality is compromised.

Software Configuration Management (SCM)

Configuration management is the development and use of standards and procedures for managing an evolving software system (Sommerville 2006). Pressman (2000) describes SCM as a set of activities designed to control change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products, controlling the changes imposed, and auditing and reporting on the changes made.

Risk Management

Risk Management involves anticipating risks that might affect the project schedule or the quality of the software being developed and taking action to avoid these risks (Hall 1998). Activities that occur in risk management are risk identification, risk analysis, risk planning and risk monitoring.

2.2 Investigating The Core Practices of XP

According to Beck (2005), a leading figure of XP, there are primary and corollary practices that XP enforces. By researching work by other members of the agile alliance (Kent Beck et al. 2001), there are thirteen practices that are most commonly referred to in their work. They are; The Whole Team, The Planning Game, Small Releases, Client Tests, Simple Design, Pair Programming, Test Driven Development, Design Improvement (Refactoring), Continuous Integration, Collective Code Ownership, Coding Standards, Metaphors and Sustainable Pace (Beck 2005, Jeffries 2001). In this report each practice has been grouped into one of four areas to show relationships between practices. The four groups are Feedback, Continuous Process, Shared Understanding and Programmer Welfare. In this section, these thirteen practices are discussed in relation to the quality factors that are incorporated into the XP process as well as their coverage in relation to generic software engineering activities.

Feedback

2.2.1 The Whole Team

All the contributors to an XP project come together, to form one complete team. The team will include:

- A business representative/client who provides the requirements and sets priorities. It is important that the business representative is an end user who knows the domain and has experience of similar systems.
- Analysts who extract client information and needs to define the requirements of the system.
- Developers that will design and implement the project and may include testers, who help the client define the client acceptance tests.

- Manager(s) who keep the project on schedule and facilitate the process. They may also provide resources, handle external communications and coordinate project activities.

Quality

This practice involves the client throughout all stages of the project life increasing communication throughout development as discussed by Juric (2000). This allows the team to build the project based on an end user's perspective rather than on their own ideas of what the system shall incorporate. The quality of the project is enhanced as it is more likely that an end product will be built that meets the requirements of the client as they are involved throughout. The client and analysts will produce requirements which will allow the developers to build an initial piece of working software that can be tested and shown to the client. Beck (2005) believes this is important in incorporating quality throughout the process as the feedback from the client can then be included into the next iteration thus developing based on the client's requirements.

High Level Relation to a Standard Software Engineering Process

This XP practice does not relate to a specific generic development activity as it is more of a general guideline that emphasises all members working together as one team instead of individual units. Juric (2000) explains that one of the main characteristics of XP is to increase communication throughout the team which involves activities of guiding the project team and making sure that everyone in the team knows exactly how and what they need to do to contribute to the project. This is often performed through regular meetings that provide the opportunity to understand who is performing what task and to offer support if a team member has a problem.

2.2.2 The Planning Game / Planning Process

Each iteration begins with the planning game, an informal process that sets the agenda for the iteration. The XP planning process allows the client to define the business value of desired features and functionality and uses cost estimates provided by developers. This allows the analysts and client to prioritise requirements in the iteration (Beck 2005).

The client defines requirements with the aid of storyboards and use cases. The analysts in collaboration with the client, separate the stories into specific tasks. The developers set time estimates on the tasks and based on the time estimates given, the user stories are prioritised by the client and analysts (Kahkonen 2003). This stage of the planning game is called Release Planning. The developers are then assigned the tasks and the development section of the iteration begins which is the second section of the planning game, called Iteration planning. The team meets approximately every two weeks and makes changes to the plan depending on any problems that have arisen during the previous two weeks iteration.

Quality

This practice allows for cheap, easy and frequent feedback to add resilience to the project discusses Jeffries (2001). The faster the requirements are discovered and changes identified, the faster the project can proceed in the right direction. Juric (2000) explains that XP encourages code production before having an overall investigation of all major system's use cases and domain models which adds to the fast and frequent releases.

Quality is incorporated into the process through this practice as the client and analysts are verifying and validating requirements during each iteration of the release planning section. By re-visiting requirements, quality can be improved as there is a greater probability that the client will receive a product that meets their well defined requirements. After each iteration of the planning section, the client is shown a piece of working software that can be used to gain feedback. This feedback can then be incorporated into the next iteration.

High Level Relation to a Standard Software Engineering Process

This XP practice relates to the software requirements and specification and project management activities of the standard software engineering process. The client and analysts define requirements by use of storyboarding and use cases (Grenning 2001) with the developers time estimating and prioritising the requirements thus project planning and requirements analysis activities are involved.

2.2.3 Small Releases

Each iteration, the team releases working, tested software, delivering the business value and fulfilling the highest prioritised requirements chosen by the client. The client can evaluate the software or even allow end users to operate the software within a pilot project. As stated by Jefferies (2001), the most important aspect of releasing working, tested software and frequent deliveries is that the client can see that progress is being made.

Quality

Small releases incorporate quality as the different versions are kept reliable by XP's emphasis on testing as described later in the testing practice. By involving the clients at each release, feedback of improvements are incorporated into the next release, thus quality is built into each new evolving version of the project. Kahkonen (2003) explains that by releasing in small developments, the client's pre-existing understanding about what they want may lead to a new articulation of his needs as they are shown what functionality they will receive. The frequent releases allow reviews to be completed earlier than traditional processes and emphasise XP's characteristic of being open to change with changes being performed earlier rather than later.

High Level Relation to a Standard Software Engineering Process

Small releases can be related to validation and verification and risk management activities. By allowing the client to sample a part of the end product, the client can have their input on whether the right product is being built and answering Boehm's (Boehm 1984) question of whether the development team are building in the correct manner. Risk management activities are related as XP uses one to four week iterations, implementing priority features first which helps to address risks immediately rather than later in the project (Juric 2000).

2.2.4 Client Tests

This practice involves the client defining automated acceptance tests for each desired feature. These tests act as proof to the developers and client that the feature is operating correctly (Jeffries 2001). These tests are then automated and kept running thereafter throughout the project as they are deemed correct.

Quality

Quality is incorporated throughout this practice as the client is constantly involved by providing ideas on how to test their desired features. The acceptance tests written by the client accumulate explicit knowledge about the desired functionality. Kahkonen (2003) states that by running the acceptance tests, the team members not only enhance their understanding of the customer's expectations, but also the conformance of their implementation to the expectations.

Martin (Martin 2004) believes that continuous testing improves and maintains the quality of the code and by having a large amount of client involvement, this will also increase the probability that the developers will produce software that matches the client's requirements.

High Level Relation to a Standard Software Engineering Process

The client tests practice relates to validation and verification as unit tests are written before source code, automated and ran frequently, and the customer writes acceptance tests. These continuous client tests include validation and verification activities into the development process. Performing these tests frequently checks that the software conforms to its specification highlights Beck (2005) and that the product has a higher probability of operating in the client's expected manner.

Continuous Process

2.2.5 Simple Design

This practice approaches software design with the, 'simple is best' attitude (Jeffries 2001). An XP team designs only for the current functionality and does not include

anything that will not be of use. The design process is continuous throughout the life of the project with design steps in release planning and iteration planning. An incremental / iterative process succeeds on a good design states Kahkonen (2003) which is why XP has a focus on design throughout the project life.

Quality

The simple design practice incorporates quality as refactoring and test driven development is performed in each small increment, sometimes a daily increment, which gradually builds quality into the design. By reviewing each increment / iteration, client focus is again enforced by XP discusses Martin, (Martin 2004) resulting in creative feedback that can be used to further improve and develop on the evolving design.

High Level Relation to a Standard Software Engineering Process

This practice relates to the design activities of a standard software engineering process. The design in XP is constantly evolving due to the approach of XP embracing change rather than preventing re-work as discussed by Fowler et al (1999). Although less time is spent on design, testing prior to coding and the technique of refactoring allow a more complete design to emerge after each iteration or increment. The simple design practice includes activities which are common in the design activity of a generic engineering process such as designing a solution from the given set of requirements.

2.2.6 Pair Programming

Programmers in pairs build all production code in XP. One programmer is in control of the coding while the other focuses on the overall project aim and continually reviews the work completed by the other programmer (Beck 2005). The roles are reversed frequently in order for the pair to understand the code and to become familiar with the whole system. This practice also enhances team-wide communication due to working in a close partnership.

Quality

This practice has been investigated and has shown there are benefits to pair programming which incorporates quality into the process. Research performed by Canfora (2007) into pair programming has shown that there is an increase in team communication. This builds on the XP whole team practice where there is more likelihood that the developers understand the overall project aim as well as the technical aspect of coding. Another study by Dybå et al (2007) showed that for complex programming tasks, having another programmer contribute their ideas resulted in achieving a more accurate, complete solution enhancing the quality of the design and actual code of the software. Simpler tasks were also found to be completed quicker which is a contributing factor to XP's fast software development.

Pair Programming, in addition to providing more complete coding solutions and quicker development times, also serves to communicate knowledge throughout the team. Tomek (2007), a teacher of XP, states that as pairs switch roles, they benefit from learning new skills and knowledge from one another thus becoming more valuable to their team and company.

High Level Relation to a Standard Software Engineering Process

Pair programming in relation to a generic software engineering process has similar concepts to implementation, and validation and verification activities. As this XP practice is the physical coding of the requirements and design, it incorporates implementation activities. Validation and verification is enforced as programming is performed in pairs which lead to the developers explicitly articulating their thoughts so the other member can understand the code. By programming in pairs, it acts as a safety precaution where they can check each others' code claims Kahkonen (2003).

2.2.7 Test Driven Development

Feedback is critical to XP and in software development, Jefferies (2001) believes that good feedback requires good testing. Each new piece of functionality begins with writing a test before coding starts. This allows feedback to be received early in development allowing the developer to analyse their progress. By testing before coding, the developer is forced to build in increments, building one test at a time.

Quality

The test driven development practice incorporates quality as it acts as a safety precaution to the developer as they know if a change they have made to the code has passed or failed due to the test. Sangwan (2006) discusses that after each new piece of functionality is added to the software, all tests are re-run to check compatibility between each piece of functionality, which improves code quality by identifying breaking points early in the development.

The developer is forced to focus on the user requirements due to writing the tests upfront. This means that an overall level of understanding is achieved in what the client requires and again involves the client in another stage of the project increasing communication throughout the whole team.

Xing (2006) highlights the importance of refactoring as it removes duplication between the test code and the production code thus forcing the code to be of a good design. As refactoring is performed at each increment, the evolving design is continuously updated and being re-worked which inevitable builds quality into the design.

High Level Relation to a Standard Software Engineering Process

The test driven development practice relates to validation and verification as activities such as unit testing and acceptance testing are performed. These tests in both the generic development activity and XP practice check that the software conforms to its specification and that the product meets the client's needs by operating in the expected manner.

2.2.8 Design Improvement (Refactoring)

The developer uses the refactoring technique throughout each increment, which Fowler et al (1999) describes as a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior. Refactoring allows the developer to re-run test cases frequently without damaging existing functionality and aids in removing duplication of code, which is a sign of poor design. Ultimately, the goal of this practice is to increase cohesion of the code, while lowering the coupling, which is recognised as a main feature of good software design as emphasised by McConnell (1996).

The practices of design improvement and test driven development strongly support one another as test driven development checks that the changes to the code have not created errors. These XP practices are more effective when used together rather than only one of either being implemented in a project claims Jeffries (2001).

Quality

This practice incorporates quality into the project as it focuses on creating a high-quality design. By removing duplication, a feature of poor design as described by Fowler et al (1999), this practice used in each iteration / increment promotes a simpler design, which is easier to understand and increases cohesion whilst reducing coupling building quality into a gradual design.

High Level Relation to a Standard Software Engineering Process

This practice relates to the design phase of a standard software engineering process. It involves activities such as refactoring to improve design and rework to create an evolving fuller design during each increment. Although the design is evolving, which is encouraged by Agile methodologies, activities such as refactoring are used in traditional development processes thus this practice relates to the generic software development activity of design.

Shared Understanding

2.2.9 Continuous Integration

The development team should always be working on the latest version of the software. Since different team members may have versions saved locally with various changes and improvements, they should try to upload their current version to a code repository every few hours, to try to keep the system fully integrated. Continuous integration will avoid delays later on in the project cycle, caused by integration problems claims Beck (2005).

Quality

This practice incorporates quality into the process as when each developer builds and uploads their code, it reduces the risk of longer-term integration problems. The code is analysed on each upload to the repository thus is kept up-to-date and continuously checked for errors. By building and integrating the code several times daily, bugs and errors that appear at integration time are reduced and the code quality is continuously kept to a high standard.

High Level Relation to a Standard Software Engineering Process

Continuous integration relates to configuration management activities as code is uploaded to a repository and integrated with other code whilst recording takes place of the changes being made. The continuous integration process helps the team members to keep their work aligned (Kahkonen 2003). Configuration management activities allow for recording, tracking and authorising changes in a controlled framework which XP encourages in order to continuously integrate new coding with old.

2.2.10 Collective Code Ownership

Collective code ownership means that everybody is responsible for all the code and everyone has the privileges to make changes to it if necessary. This practice works in conjunction with pair programming as by working in different pairs, all the programmers get access to all parts of the code.

Quality

This practice incorporates quality as the code gets the benefit of many developers' specialised attention, which increases code quality and reduces defects.

Collective code ownership could cause problems if someone was working on code that they did not understand. The two XP practices of test driven development and pair programming help to prevent this problem states Beck (2005). Test driven

development acts as a safety net as when the developer changes the code, the initial test before coding is ran to check if the altered code will pass the test.

High Level Relation to a Standard Software Engineering Process

This practice does not relate to any of the generic activities as it relies on other practices to be effective. It encourages the developers to take responsibility for all the code, if they want to make a change, they can do so. This practice relies on test driven development and pair programming and would not be practical to implement on its own thus cannot be grouped under a general software engineering activity.

2.2.11 Coding Standards

XP teams follow a common coding standard, so that all the code in the project looks as if it was written by a single individual. The specifics of the standard are not important: what is important is that all the code looks familiar, in support of the collective ownership practice.

Quality

The coding standards practice does link to quality as all coding is written in one style avoiding confusion when other developers are trying to understand or alter the code. This allows for a simpler integration when bringing code together (Jeffries 2001), written by separate developers thus coding quality is kept to the same high standard.

High Level Relation to a Standard Software Engineering Process

This practice relates to software quality assurance activities as the code is kept to an industry or organisational recognised standard, which is used throughout the project life. Huo (2004) believes this forces quality assurance on to the developer whilst coding to establish a general high standard of coding throughout the project. This means that code written by many developers can be read as though one individual wrote it. This increases understanding and makes analysing the code simpler than having to understand different developers' styles of coding.

2.2.12 Metaphors

The metaphor practice is a naming concept for classes, methods and patterns that shape the core flow of the system being built describes Huo (2004). From the name only, the team should know the functionality of a particular class/method. XP uses a common system of names to allow the development team to understand the functionality and to know where to look when modifying code.

Quality

The metaphors practice incorporates quality as it allows the development team to plan, design and develop the project based on metaphors that mean the same to all members of the team. Jeffries (2001) understands that this helps when planning, designing and developing as it provides a matching level of understanding throughout the whole team.

High Level Relation to a Standard Software Engineering Process

This practice directly relates to coding in the implementation phase of a standard engineering process as metaphors when coding emphasise a generic naming convention so other programmers within the team can understand the coding. Thus it is more of a communication tool / technique which aids the implementation phase and cannot be added to the experiment. Huo (2004) explains that the metaphor at the implementation stage bridges the gap between developers and users to ensure an easier time in discussion and allows the developers to use examples through an effective naming convention. During implementation activities, the metaphor practice also contributes to the team's development of software architecture.

Programmer Welfare

2.2.13 Sustainable Pace

This practice introduces the concept that software developers should not work more than 40-hour weeks and that overtime should be kept to a minimum. Since the development cycles are short cycles of continuous integration, and full development (release) cycles are more frequent, the projects in XP do not follow the typical crunch time that other projects require, reducing the need for overtime. Also included in this concept is that people perform best and most creatively if they are rested.

Quality

The practice of sustainable pace deals more with personal issues in XP rather than on quality factors.

High Level Relation to a Standard Software Engineering Process

This practice does not relate to any of the generic activities of a standard development process as it is concerned with employee welfare rather than quality of development.

2.3 Investigating The Process Areas of The CMMI-Dev

In this section, there will be analysis and discussion of the CMMI for Development's twenty two process areas, their link to quality, and their coverage in relation to generic software engineering activities. Each process area is defined by a set of specific goals and generic goals with associated specific practices and generic practices. A goal, within any process area is a desirable state that should be attained by an organisation. To achieve the goal, the organisation must complete the associated 'specific practices by goal' and 'generic practices by goal'. Figure 1 below shows the breakdown of the process area.

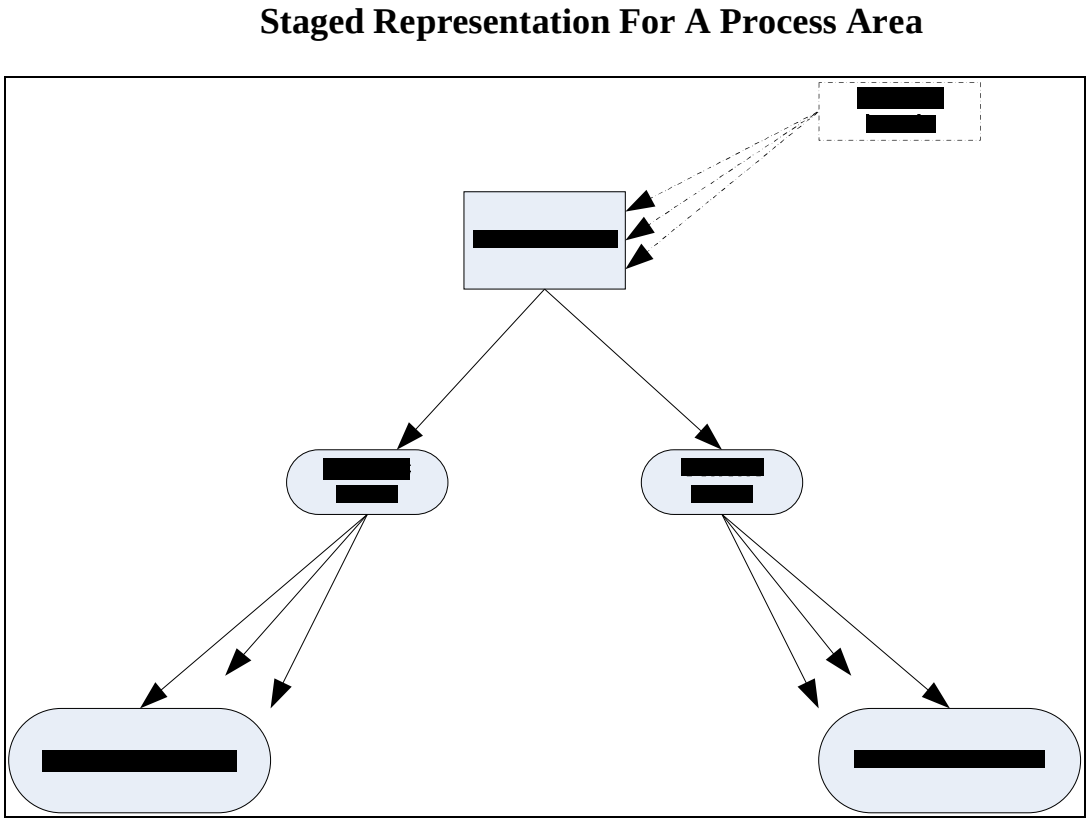


Figure 1: Staged Representation For A Process Area (adopted from CMMI Product Team 2006)

The CMMI for Development consists of twenty two process areas which are grouped into four categories by the CMMI-Dev as shown in Figure 2. They are Process Management, Project Management, Engineering and Support. In Figure 3, it shows the staged CMMI-Dev maturity level with each accompanying process area.

CMMI-Dev Categories

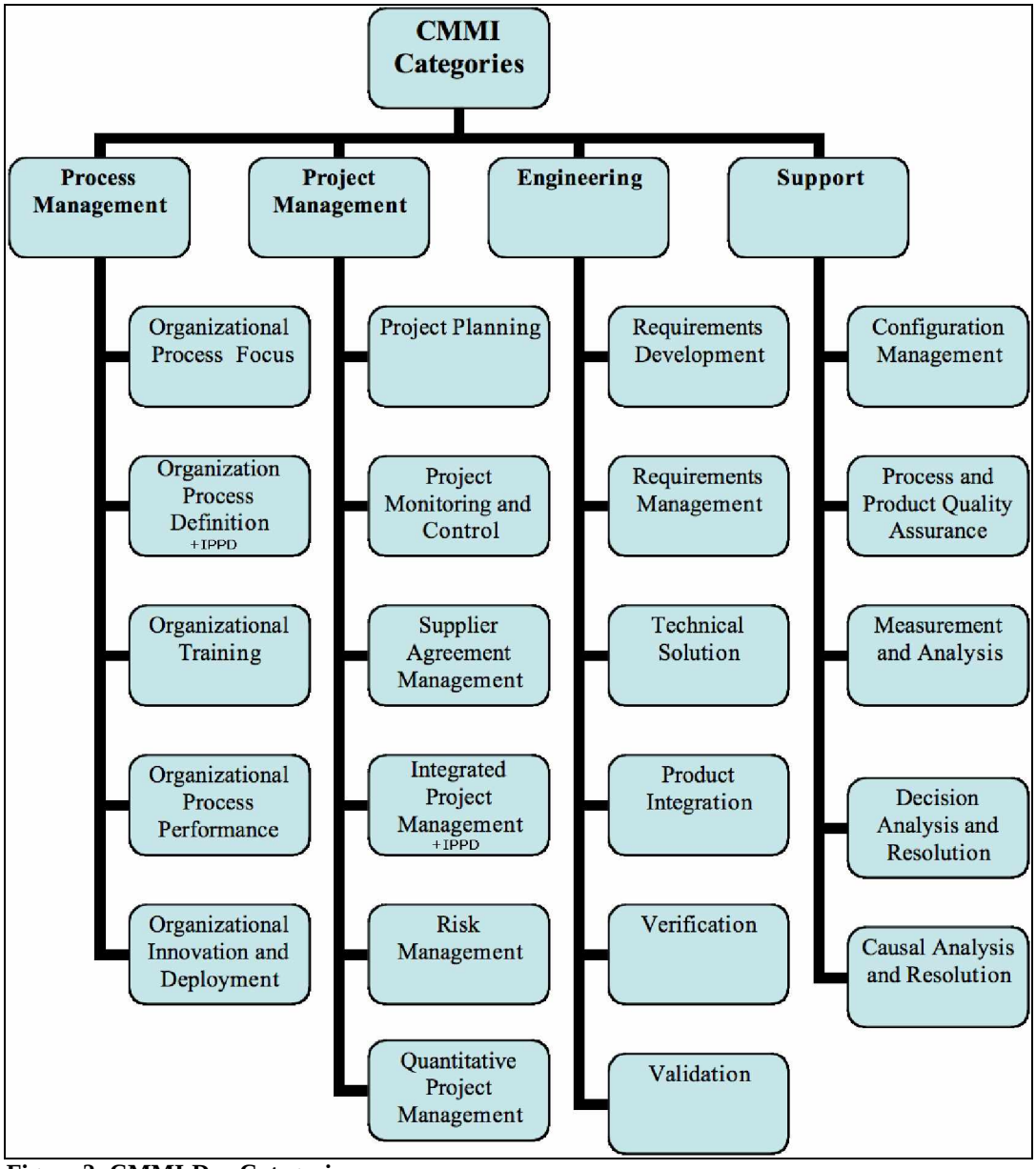


Figure 2: CMMI-Dev Categories

CMMI-Dev Staged Representation

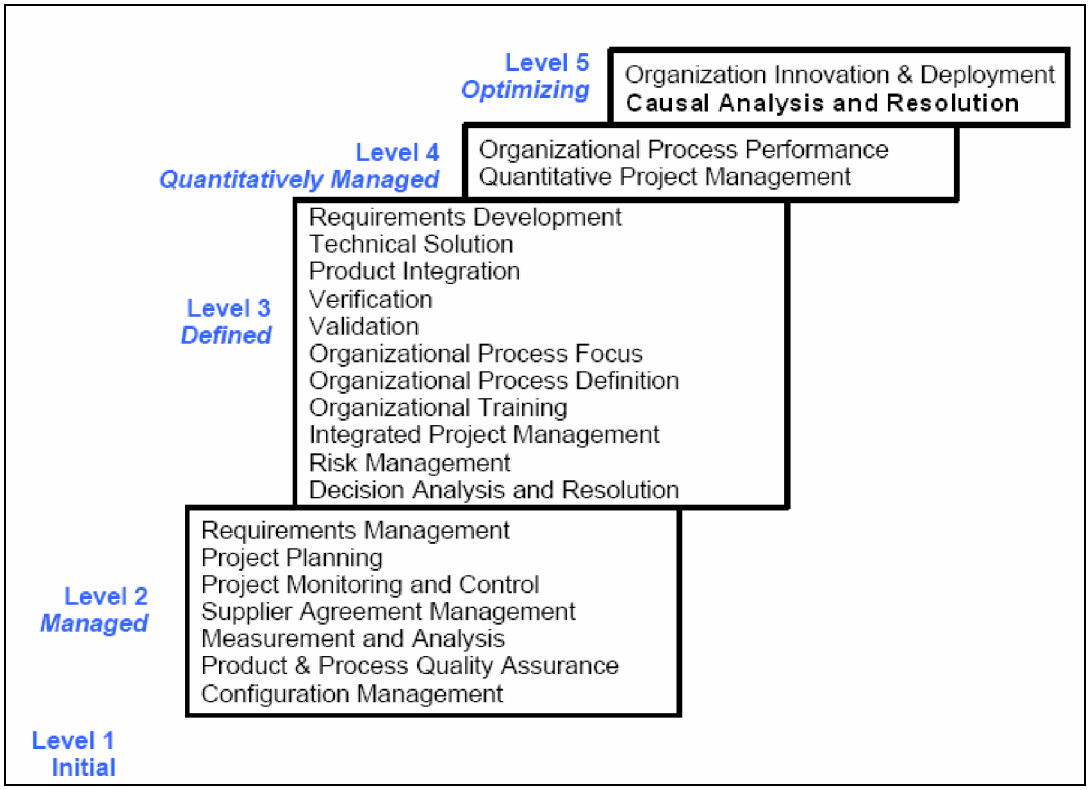


Figure 3: CMMI Staged Representation (Norausky 2002)

This study will analyse the CMMI-Dev staged representation as shown in Figure 3. CMMI-Dev can be appraised using two different approaches: staged and continuous. The staged approach yields appraisal results as one of five maturity levels. The continuous approach yields one of six capability levels. The differences in these approaches are felt only in the appraisal; the best practices are equivalent and result in equivalent process improvement results state the CMMI Product Team (2006).

The following sections analyse and discus the CMMI for Development's twenty-two process areas, their link to quality, and their coverage in relation to generic software engineering activities.

2.3.1 Casual Analysis and Resolution

Purpose

The CMMI-Dev v1.2 official document (2006) emphasises that the purpose of Causal Analysis and Resolution is to identify causes of defects and to take action to prevent them from occurring in the future.

Quality

Causal analysis and resolution improves quality and productivity by preventing the introduction of defects into a product. Beck (2005) emphasises the reliance on detecting defects early in the lifecycle to be more cost effective than trying to fix those detected towards the end of the project at the testing stage. It is more effective to prevent defects from being introduced by integrating causal analysis and resolution activities into each phase of the project, which incorporates quality at each stage.

Communication is a key aspect of the casual analysis and resolution process area. It encourages communication between developers, management and the client. This will have a direct effect on the likelihood that the client will receive a product that matches their needs.

High Level Relation to a Standard Software Engineering Process

Casual analysis and resolution can be related to the quality management activities of a standard software engineering process emphasising communication and trying to reduce defects throughout. As defects and problems are encountered in all projects or even at earlier stages in the same project, casual analysis and resolution activities can be used to communicate lessons learnt across the project to find a solution.

Norausky (2002) emphasises the quality management approach of casual analysis and resolution's practices that pro-actively focus on involving the client throughout the project. Another practice harnesses employee project experience to determine defect similarities and the organisational shift in focus from discovery of defects to prevention.

2.3.2 Configuration Management

Purpose

The Institute of Configuration Management (CM) (2008) describes CM as a method that enables an organisation to accommodate change and keep requirements clear, concise and valid. The purpose of CM is to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits.

Quality

The CMMI Configuration Management process area is effectively implemented at the beginning of a project and continued throughout. All configuration items are identified and those that have reached maturity are added to the configuration. Quality is incorporated as by identifying, controlling and auditing the configuration items, changes can be tracked to reveal any adverse affects on the project.

Integrity of the configuration is upheld throughout by reviewing the work products, establishing baselines and reporting the status of the configuration items (CMMI Product Team 2006). Relevant stakeholders are involved throughout this process area and are encouraged to remain active during the process. Their knowledge of the domain can benefit the overall quality of the final product as when changes are being requested, it may require their input for a decision to be achieved.

High Level Relation to a Standard Software Engineering Process

As configuration management is commenced at the beginning of the project and used throughout, it can be related to the generic activity of configuration management. Requirements analysis activities identify requirements and items that may mature and be incorporated into the configuration. Design activities will again identify items but may also require controlling any changes that are requested to the design and at the implementation, verification and validation stages, change requests will be more frequent as errors and omissions are discovered.

2.3.3 Decision Analysis and Resolution

Purpose

The purpose of Decision Analysis and Resolution (DAR) is to analyse possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria (CMMI Product Team 2006).

Quality

There is a need to develop methods and models that can be used to estimate reliability not only to demonstrate compliance with requirements but also to monitor reliability through the product life declares Hodge (2002). This process area incorporates quality based on Hodge's statement by evaluating alternative solutions to an issue against established criteria to determine the most suitable and effective solution. This means that there is a higher probability that the solution meets the demands of relevant stakeholders.

High Level Relation to a Standard Software Engineering Process

The DAR process area of CMMI, according to Chrissis, Konrad & Shrum (2006) provides a standard way for a formal evaluation process which could be used to support any process throughout the product lifecycle whenever a significant decision is to be made.

Although applicable to many areas and used throughout the product's life, this process area focuses on risk stages of a standard software engineering process. Initially the process begins by planning and creating organisational guidelines which allows evaluation criteria to be established. Risk activities include developing alternative solutions and methods that are considered and put in place when a decision is to be made (Vantakavikran 2007).

2.3.4 Integrated Project Management

Purpose

According to the CMMI-Dev document (2006), the purpose of Integrated Project Management (IPM) is to establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process that is tailored from the organisation's set of standard processes.

Quality

This process area manages stakeholder involvement and dependencies. Quality can be built into the project by monitoring plans, objectives, problems and risks that will fulfill stakeholder commitments and track and resolve co-ordination issues discusses Macrcal (2007). These activities allow the client's expectations to be met and help to produce a product that fulfills their requirements.

High Level Relation to a Standard Software Engineering Process

Integrated project management relates to the project management activities of a generic software engineering process. By involving the client and encouraging them to share their vision of the system with the team, generic project management activities and the IPM process area share a common goal to produce a product that completes the client's needs through planning, organising and monitoring tasks whilst involving the client.

2.3.5 Measurement and Analysis

Purpose

The purpose of Measurement and Analysis (MA) is to develop and sustain a measurement capability that is used by management to give the visibility and focus needed to guide the use of measurement in their process improvement efforts (CMMI Product Team 2006).

Quality

The need to focus early in a project on measurement and analysis is important as it helps provide objective insight into issues and processes, along with the ability to objectively identify and manage risks and provide early detection and resolution of problems. Goldenson, Jarzombek & Rout (2003) highlight that measurement facilitates evidence-based team communication and enables objective planning and estimating. This increases the probability of project success. By performing measurement and analysis correctly, it may provide information that improves decision making in time to affect the business or project outcome (J. McGarry et al. 2001).

High Level Relation to a Standard Software Engineering Process

Goldenson, Jarzombek & Rout (2003) of the Software Engineering Institute believes that the Measurement and Analysis process area supports all process areas but includes mainly project management activities. It provides practices that guide projects and organisations in aligning their measurement needs and objectives with a measurement approach that will provide objective results that can be used in making informed decisions.

2.3.6 Organisational Innovation and Deployment

Purpose

The purpose of Organisational Innovation and Deployment (OID) is to select and deploy incremental and innovative improvements that measurably improve the organisation's processes and technologies.

High Level Relation to a Standard Software Engineering Process

Organisational Innovation and Deployment does not relate to the standard engineering activities. This process area is concerned with innovating new ideas, to improve on current organisational practices. Chong (2006) describes innovation as the intentional introduction and application within an organisation of ideas, processes, products or procedures designed to significantly benefit that organisation. This means constantly finding ways to self improve on processes or practices regardless whether they are performing correctly.

2.3.7 Organisational Process Definition

Purpose

The purpose of Organisational Process Definition (OPD) is to establish and maintain a usable set of organisational process assets and work environment standards (CMMI Product Team 2006).

High Level Relation to a Standard Software Engineering Process

This process area does not relate to a generic software engineering activity. It is concerned with the organisation's process asset library, which is a collection of items maintained by the organisation for use, by people and projects of the organisation such as lifecycle models, process related documentation and process tailoring guidelines.

2.3.8 Organisational Process Focus

Purpose

This process areas purpose, according to Paulk (2001), is to establish organisational responsibility for software process activities that improve the organisation's overall software process capability.

High Level Relation to a Standard Software Engineering Process

This process area does not relate to a generic software engineering activity as it is concerned with organisational process assets which are used to describe, implement and improve an organisation's process (Wu, Wang & Fang 2007).

2.3.9 Organisational Process Performance

Purpose

The purpose of Organisational Process Performance (OPP) is to establish and maintain a quantitative understanding of the performance of the organisation's set of standard processes in support of quality and process-performance objectives, and to provide the process performance data, baselines, and models to quantitatively manage the organisation's projects (CMMI Product Team 2006).

Quality

The Organisational Process Performance process area directly relates to quality by using metrics to measure the actual results of following a process. The common measures for the organisation are composed of process and product measures that can be used to summarise the actual performance of processes in individual projects in the organisation. These measurements can directly reflect quality (Huang, Han 2006). By monitoring the actual performance within an organisation, weaker aspects can be improved to contribute to a higher standard of quality.

High Level Relation to a Standard Software Engineering Process

This process area does not relate to the generic standard engineering activities as it regards understanding, in a quantitative manner, the performance of the organisation's set of standard processes which is not included in a standard software development project.

2.3.10 Organisational Training

Purpose

The purpose of Organisational Training (OT) is to develop the skills and knowledge of people so they can perform their roles effectively and efficiently discusses Paulk (2001).

High Level Relation to a Standard Software Engineering Process

This process area does not relate to the generic standard engineering activities as it regards meeting tactical training needs identified by individual projects and support groups within an organisation.

2.3.11 Process and Product Quality Assurance

Purpose

The purpose of Process and Product Quality Assurance (PPQA) is to provide staff and management with objective insight into processes and associated work products (CMMI Product Team 2006).

Quality

As software is increasingly becoming a larger part of many products and services, and the quality of a system is highly influenced by the quality of the process (Software Engineering Institute 2007), a well-defined software development process plays a big part in the quality of a system. The quality of a product could be increased due to this process area as Wang's (2007) study tries to prove. The study emphasises that each team member must objectively evaluate the adherence of the performed processes, associated configuration items and services to applicable process descriptions, standards and procedures to identify and document non-compliance issues.

High Level Relation to a Standard Software Engineering Process

The quality assurance activities within the Process and Product Quality Assurance process area support the delivery of high-quality products and services. By concentrating on quality assurance practices, it provides the project staff and managers at all levels with appropriate visibility into, and feedback on, processes and associated work products throughout the life of the project. This process area directly links with the SQA activities of a general software engineering process.

2.3.12 Product Integration

Purpose

The purpose of Product Integration (PI) is to assemble the product from the product components, ensure that the product, as integrated, functions properly, and deliver the product (CMMI Product Team 2006).

Quality

Quality is incorporated by this process area as it is not just a one time assembly of components into an end product, but in stages of an iterative / incremental fashion which allows customer testing on part of the system (Chrissis, Konrad & Shrum 2006). In each successive integration, prototypes can be constructed, evaluated, improved and then reconstructed to a higher quality based on the knowledge gained in the evaluation process.

High Level Relation to a Standard Software Engineering Process

This process area relates to the implementation phase of a generic software engineering process. This process area involves integrating components to form the full product.

2.3.13 Project Monitoring and Control

Purpose

The CMMI-Dev document (CMMI Product Team 2006) states the purpose of Project Monitoring and Control (PMC) is to provide an understanding of the project's progress so that appropriate corrective actions can be taken when the project's performance deviates significantly from the plan.

Quality

Project progress is primarily determined by comparing actual work product and task attributes, effort, cost, and schedule. Quality is determined by comparing these attributes to the project plans at prescribed management milestones or control levels within the project schedule. By having these metrics, quality can be made visible which enables timely corrective action to be taken when performance lowers significantly from the plan. If an attribute such as cost or effort was to be having a negative influence then project quality would be compromised.

High Level Relation to a Standard Software Engineering Process

This process area relates to project management as monitoring metrics are used by the manager to control the progress of the project. The manager can administer project resources, cope with problems and direct staff dependant on effort and work metrics (Pressman 2000).

2.3.14 Project Planning

Purpose

Sommerville (2006) describes project planning as a framework that enables management to make responsible estimates of resources, cost and schedule when defining project activities.

Quality

Project planning is a key aspect of project management. The responses from 154 experienced software project developers who contributed to a recent study by Jiang et al (2004), confirmed that software process management maturity is positively associated with project performance. To software managers and process improvement teams, the results of this study suggest that project planning could be a useful guide to improving their current state of software processes in order to improve project and software quality.

High Level Relation to a Standard Software Engineering Process

Project planning is used throughout the project lifecycle. Many initial plans will be revised as the project progresses to address changes in requirements, inaccurate estimates, corrective actions, and process changes. The project management process of a standard software engineering process will contain activities that include project planning which may discover a relationship between these goals and activities. Project planning also includes preparing and analysing risks, which may include activities of risk management. This process area contains practices that relate to the generic activities of project management.

2.3.15 Quantitative Project Management

Purpose

The purpose of Quantitative Project Management (QPM) is to quantitatively manage the project's defined process to achieve the project's established quality and process-performance objectives (CMMI Product Team 2006).

Quality

The Quantitative Project Management process area directly relates to quality by using metrics to analyse if the project's performance and quality objectives are being met. If the metrics determine there is fluctuation in performance then corrective action can be taken. By monitoring performance metrics there is a greater chance that the project will meet its quality objectives discusses McGarry (2001).

High Level Relation to a Standard Software Engineering Process

This process area relates to quality assurance activities but cannot be applied to a standard software engineering process. The organisation must have a defined process already established to apply quantitative project management. The defined process used to develop software uses continuous monitoring of metrics such as work effort, man hours per project, cost per employee and determining metrics such as lines of code and error occurrence, which relates to software quality assurance. These metrics are used to monitor project resources and can highlight problems with the project when a metric varies from the expected.

2.3.16 Requirements Development

Purpose

The purpose of Requirements Development (RD) is to produce and analyse client requirements, product requirements and product component requirements describes Chrissis, Konrad & Shrum (2006).

Quality

All projects have requirements thus it is critical to be able to define, understand and document requirements fully. By working in conjunction with the client and understanding their needs, expectations and constraints, an understanding will be developed of what will satisfy stakeholders' needs (CMMI Product Team 2006). Quality can then be built into the product as a clear definition of client requirements is understood and can be used to address product and product component requirements.

High Level Relation to a Standard Software Engineering Process

The standard systems engineering process for creating a system design is normally decision-rich claims Buede (1997). Buede (1997), in a separate paper, states that the analyst normally completes a great deal of analysis and experience to find a very good solution that satisfies all of the mandatory requirements of the stakeholders and delivers as much performance as possible within the guidelines of cost and schedule. The relation to a standard software engineering activity would be in the section of software requirements and specifications.

2.3.17 Requirements Management

Purpose

The purpose of Requirements Management (REQM) is to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products (CMMI Product Team 2006).

Quality

In an empirical evaluation by Damian (2002) which focuses on process improvement, indicates that there are benefits as well as perceived long-term advantages to design and testing from a good requirements management process area. Quality is improved, as findings suggest that a thorough requirements analysis results in a more clearly defined, better understood and specified requirements. This enhances the ability to address the market needs and product strategy requirements of the organisation.

High Level Relation to a Standard Software Engineering Process

The requirements management process area contains activities that are familiar to the software requirements and analysis section of a general software engineering activity. The management aspect of this area is to document requirements changes and rationale and to maintain bidirectional traceability between source requirements and all product and product component requirements (CMMI Product Team 2006). Figure 4 shows activities, objectives and deliverables which are common to standard software requirements and analysis activities.

Requirements Management Diagram

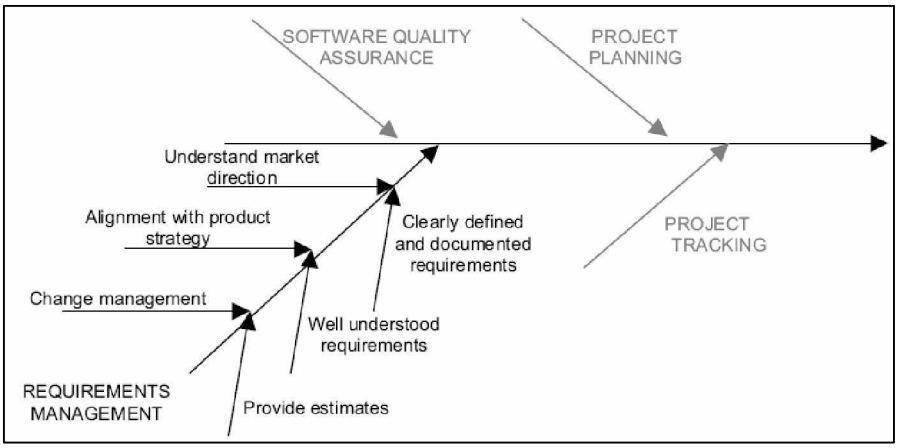


Figure 4: Requirements Management Diagram (Damian 2002)

2.3.18 Risk Management

Purpose

Ould (1999) claims that the purpose of risk management is to manage, anticipate and avoid risks that may have an effect on project resources such as budget or schedule and affect quality of the software being developed.

Quality

Risk management can help prevent quality of the software from deteriorating as it involves continuous monitoring of the project, which is an important part of management (Sommerville 2006). A continuous risk management approach should be applied as it effectively anticipates and mitigates risks that may have a critical impact on the project thus reducing the risk of project quality being affected.

High Level Relation to a Standard Software Engineering Process

The generic risk management activities of a standard software engineering practice relates to the risk management process area of the CMMI-Dev as both address and monitor issues that could endanger the achievement of critical objectives.

2.3.19 Supplier Agreement Management

Purpose

The purpose of Supplier Agreement Management (SAM) is to manage the acquisition of products from suppliers (CMMI Product Team 2006).

High Level Relation to a Standard Software Engineering Process

Supplier agreement management does not relate to any of the generic engineering activities. This process area primarily addresses the acquisition of products and product components that are delivered to the project's client under a formal agreement. This is established to manage the relationship between the organisation and the supplier.

2.3.20 Technical Solution

Purpose

The purpose of Technical Solution (TS) is to design, develop, and implement solutions to requirements. Solutions, designs, and implementations encompass products, product components, and product-related lifecycle processes either singly or in combination as appropriate (CMMI Product Team 2006).

Quality

Technical solution can improve quality as it encourages evolutionary design states Royce and Walker (2002). Developers are forced to build in stages or in an iterative manner which builds more quality into the design at each level. The process area focuses on design approaches that may potentially satisfy an appropriate set of allocated requirements and includes developing detailed designs for selected solutions. This incorporates quality as all the information needed to manufacture or code a solution is refined, completed and corrected by the activities in the technical solution process area before the final design is cleared for implementation.

High Level Relation to a Standard Software Engineering Process

This process area may relate to the design section of a standard software engineering process. Activities such as considering alternate solutions, designing the solution and providing design documentation are common activities.

2.3.21 Validation & Verification

Purpose

The purpose of Verification is to ensure that selected work products meet their specification where as the purpose of Validation is to demonstrate that a work product or product component fulfills its intended use as discussed by Sommerville (2006).

Quality

Validation & Verification (V&V) incorporate quality into software throughout the process of software engineering as proper application of methods and testing tools, formal technical reviews, solid management and measurement all lead to quality that is confirmed during testing discusses Pressman (2000).

Quality is built into software by performing peer reviews and inspecting documents such as the requirements documents, design models and program source code. This should be performed throughout the project lifecycle where errors and omissions are found and quality assessed continuously instead of only at the testing phase.

High Level Relation to a Standard Software Engineering Process

The verification and validation process areas of the CMMI-Dev framework comprises of similar activities as the verification and validation of a generic software engineering process. Activities that encompass software quality assurance activities are quality and configuration audits, performance monitoring, feasibility studies, algorithm analysis, development testing and installation testing among others. Verification and validation incorporates quality assurance principles when audits are performed as it provides conformity between documents whilst deliverables comply with the requested changes.

2.4 Literature Review Conclusion

This literature review discusses and evaluates Extreme Programming (XP) practices and CMMI-Dev process areas in terms of quality and the relations they have to generic software development activities. The literature review has satisfied the aims and objectives set by providing an insight into Extreme Programming and the CMMI-Dev quality issues that surround their practices and process areas.

The literature review has revealed which practices and process areas of the CMMI-Dev and XP have common activities with a standard development process. It also revealed those practices and process areas that do apply to generic development activities and were not included in the experiment. Overall there are fifth-teen CMMI-Dev process areas and eleven XP practices assessed in the experiment. The process areas and practices that were not included in the experiment are listed below in Figure 5. They are displayed in a clear format which summarises why they were not chosen to be included in the experiment.

Process Areas / Practices That Do Not Apply

Methodology	Process Area / Practice	Reason
CMMI-Dev	Organisational Innovation and Deployment	This process area is concerned with innovating new ideas, to improve on current organisational practices.
CMMI-Dev	Organisational Process Definition	It is concerned with the organisation's process asset library, which is a collection of items maintained by the organisation for use by people and projects.
CMMI-Dev	Organisational Process Focus	Deals with organisational process assets which are used to describe, implement and improve an organisation's process.
CMMI-Dev	Organisational Process Performance	Concerned with quantitatively understanding the performance of the organisation's set of standard processes.

CMMI-Dev	Organisational Training	Regards training needs identified by individual projects and support groups within an organisation.
CMMI-Dev	Process and Product Quality Assurance	Deals with quantitatively managing the project's defined process.
CMMI-Dev	Supplier Agreement Management	This process area addresses the acquisition of products and product components that are delivered to the client.
XP	Collective Code Ownership	This practice relies on other practices to be effective and thus cannot be analysed individually.
XP	Metaphors	This practice is more of a communication technique that aids the implementation phase.
XP	Sustainable Pace	This practice is concerned with employee welfare rather than quality of software development.
XP	The Whole Team	This practice is more of a general guideline that emphasises all members working together as one team instead of individual units.

Figure 5: Process Areas / Practices That Do Not Apply

By analysing each practice and process area and evaluating relations to general software engineering activities, the experiment was able to be conducted on a foundation of reasoning as each practice and process area was able to be grouped into a common generic software engineering activity. Figure 6 shows what CMMI-Dev process area or XP practice was grouped into the related generic activity and how many specific practices (SPs) were required to be analysed by the related XP practice.

Relationship Activity Diagram

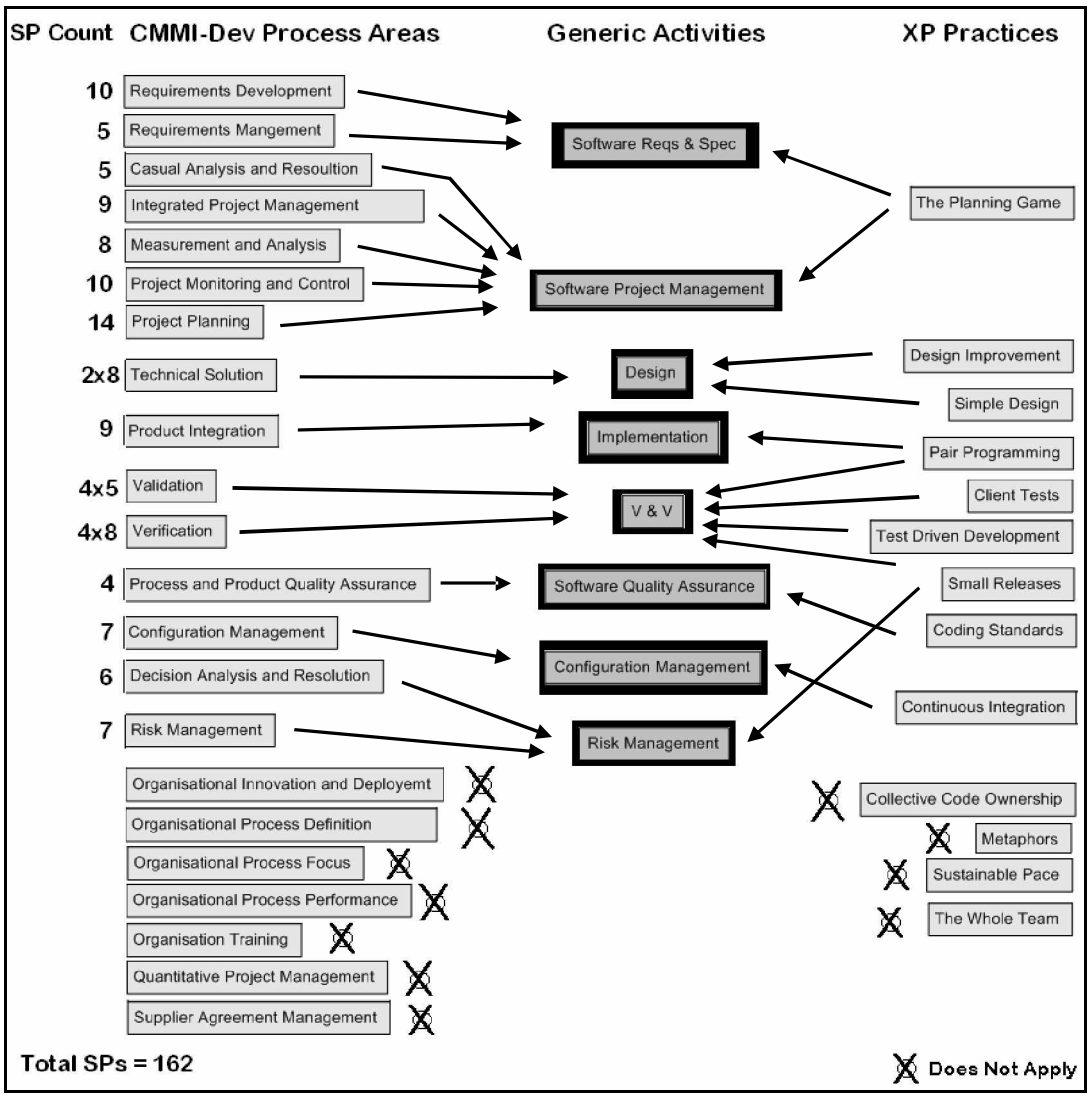


Figure 6: Relationship Activity Diagram

From the literature review, a basic conclusion of high level relationships between generic activities of a software development process, XP practices and CMMI-Dev process areas allowed the study to group process areas and practices under software development activity headings as shown in Figure 6. The experiment could now be conducted as an understanding was developed, due to the literature review, of what practices and process areas could be related under general software development activities in which relationships could be revealed.

3.0 Methodology

The purpose of this chapter was to present in detail the specifics of the primary research approach. Included is thorough discussion and critical evaluation of literature to justify why the selected research approach was chosen for the experiment.

3.1 Primary Research Method

The project involved an explanatory experiment that analysed the CMMI-Dev process area specific goals, and XP's practices to identify relationships between them. The type of primary research method used was an explanatory experiment as it enabled the study to highlight or disregard links between factors by means of theory (Oates 2006).

Oates (2006) explains that an experiment, in academic research, is a strategy that investigates the cause and effect of relationships, seeking to prove or disprove links. This explanation relates to the experiment adopted by this study, which investigates relationships between the CMMI-Dev process areas and their associability with XP's quality practices.

In the study by Marcal (2007), an explanatory experiment is successfully used to analyse the relationships between SCRUM and the project management process areas of the CMMI. The final outcome results in a mapping between SCRUM and the CMMI in which organisations / process improvement teams can clearly relate their own traditional quality management framework (QMF) practices with an Agile development process. Currently the CMMI-Dev only gives examples of processes and practices at the organisational level, but does not provide 'how to' specifics for software developers and their teams whilst XP does as discussed by Konrad & Over (2005). It would be beneficial to have a hybrid framework to satisfy both efficient processes and give reasoning to how software developers could implement them.

Stojanovic, Dahanayake & Sol (2004) use a similar explanatory method to Marcal (2007) where different Agile methods are compared to one another in a successful manner. Different Agile method characteristics are listed in tables for each Agile method and then a comparison is performed where as this experiment involves analysis between CMMI-Dev process area specific goals and XP's practices to identify whether there are relationships between them. This study adopted a similar approach based on the profound and well established explanatory experimental methods of Marcal (2007) and Stojanovic (2004). The purpose of using an explanatory experiment was to allow organisations and process improvement teams to make clear judgements and informed decisions based on the results of the explanatory experiment. They can decide on what practices of the XP Agile development process would suit their own traditional CMMI-Dev framework or vice versa.

Included in the experiment was grounded theory, which Strauss & Corbin (1990) states that pre-conceptions are inevitable. This reiterates the importance of the literature review as it assumes the researcher should already know about the topic before commencing the experiment. The grounded approach provided flexibility in providing new insight, which

is difficult to achieve when using hypotheses as it can be easy to prove what has already been discovered by other studies.

3.2 Alternative Methods

The case study approach, which involves studying a particular contemporary phenomenon within its real life context, was considered suited to this project but despite the initial authorisation of a case study to be included, the organisation withdrew from the study. Fernández (2005) regards case studies to be quite procedurally complex and have a high degree of risk even when planned properly by inexperienced researchers, which may have been a concern.

If a researcher is performing a case study on an organisation that they have a background with, then the question of bias arises in their project as they will already have an inclination of the result discusses Fernández (2005). The grounded theory and comparative analysis approach adopted in this study, is effective in countering the question of researcher bias by setting aside organisational beliefs by letting the theory produce the results.

3.3 Precise Nature of the Experiment

The mapping between CMMI process areas and XP practices considers the staged CMMI-Dev model (CMMI Product Team 2006), shown in Figure 3, page 29. For each process area, a mapping between its specific goals and the XP practices have been analysed under a generic development activity in order to fully discover relationships which reveals gaps, strengths and omissions.

An initial trial mapping was conducted as a scoping exercise in order to measure how long each CMMI-Dev process area and XP practice would take to perform the document analysis of the project. The full experiment involves a document analysis consisting of reviewing approximately 423 pages of the official CMMI-Dev document (CMMI Product Team 2006). By conducting a trial mapping, the study was able to obtain quantifiable measurements that allowed the study to calculate how many mappings could be completed within the honours project time schedule. The first generic activity was the Software Requirements and Specifications area as shown in Figure 6. Related to this 'umbrella' activity was the planning game practice of XP and the CMMI-Dev process areas of requirements development and requirements management. The trial mapping, Figure 8, shows the planning game practice and requirements development specific goals and practices with the document analysis of only specific goal one shown, Figure 9. For the full results of the requirements development and planning game mapping, see page **????** in Appendix A. Arrows show which key point of the XP practice contribute to the goal. While CMMI does not always support interpretations in an agile context, as also found by Kähkönen and Abrahamsson (2004), help can be provided by drawing up mappings describing the connections between CMMI and agile practices which is adopted in this study. The trial mapping contains the goals and practices of the CMMI-Dev process area, the XP practice, key XP points, and the rating criteria.

Rating Criteria

Rating	Criteria	
U	Unsatisfied	The process area is not addressed by XP practices.
PS	Partially Satisfied	There is some evidence that the process area is being addressed by XP practices, however the process area is not fully addressed.
S	Satisfied	The practice is fully addressed by XP practices.

Figure 7: Rating Criteria (Marcal 2007)

After analysing the relationships between practices and process areas in a generic software activity, a rating is given dependant upon the results. The rating criteria is shown in Figure 7. The rating criteria for the trial mapping are shown in Figure 9, which is for the CMMI-Dev requirements development process area and the XP planning game practice.

Trial Mapping

XP Practice – Planning Game
CMMI-Dev Process Area – Requirements Development

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria	Key Points Of Planning Game Practice
SG 1 Develop Customer Requirements SP 1.1 Elicit Needs SP 1.2 Develop the Customer Requirements **SG 2 Develop Product Requirements** SP 2.1 Establish Product and Product Component Requirements SP 2.2 Allocate Product Component Requirements SP 2.3 Identify Interface Requirements **SG 3 Analyse and Validate Requirements** SP 3.1 Establish Operational Concepts and Scenarios SP 3.2 Establish a Definition of Required Functionality SP 3.3 Analyse Requirements SP 3.4 Analyse Requirements to Achieve Balance SP 3.5 Validate Requirements	**(SG1)** Partially Satisfied **(SG2)** Unsatisfied **(SG3)** Partially Satisfied	Client presents desired features through use cases / storyboards. Developers produce cost estimates + requirement prioritisation. Initial schedule / plan produced before commencing iteration. Running software produced every 2-4 weeks. Customer involved in evaluating software. Feedback taken and incorporated into next 2-4 week iteration.

Figure 8: Trial Mapping

Supporting Document Analysis

Specific Goal 1: Develop Customer Requirements

Goal Result: Partially Satisfied

Specific Practice 1.1: Elicit Needs

Practice Result: Partially Satisfied

- This specific practice collects all project requirements by proactively identifying all additional requirements not explicitly provided by the client, unlike the planning game practice of XP as it focuses solely on the client's requirements.
- Planning game practice does not reveal all requirements of the project but only those that are important to the client such as functionality and aesthetic requirements thus the XP planning game practice only partially satisfies this CMMI-Dev specific practice.
- A sub practice of this specific practice is to engage relevant stakeholders using methods for eliciting needs, expectations, constraints, and external interfaces. The XP practice does fully fulfill this requirement as methods such as use cases and storyboards are used by the development team and client to discover and extract requirements.

Mapping Key Points:

- XP practice does not reveal all requirements where as this specific practice, to be fulfilled, needs all additional requirements stated.
- The specific practice requires engagement by the stakeholders, which the XP practice fully satisfies as the client is continually involved.

Specific Practice 1.2: Develop The Customer Requirements

Practice Result: Partially Satisfied

- Documentation is not a priority of the planning game practice as working software is regarded more important thus does not fully satisfy the specific practice although some documentation may be produced.
- The development team in the planning game practice do produce cost estimates and prioritise requirements, which partially satisfies this specific practice.
- Conflicts between requirements are to be identified and resolved by this specific practice, which the planning game practice fully satisfies as the development team prioritise client requirements, identifying the most important to the project and disregarding any that are non-relevant or unrealistic.

Mapping Key Points:

- The XP practice does not produce all documentation where as the specific practice needs all specified and documented.
- The XP practice prioritises requirements eliminating conflicts and missing information that may have been accrued during the gathering of the requirements.

Figure 9: Supporting Document Analysis

The trial mapping revealed that it approximately takes 3 hours and 10 minutes to perform the document analysis, which includes mapping the CMMI-Dev process area with an XP practice and documenting the results. This trial mapping revealed that the study was far to ambitious to be fully completed within the honours project time frame. As identified by the literature review and shown in Figure 6 on page 44, the relationship diagram, there are 22 mappings with 162 specific practices of the CMMI-Dev that could be analysed but due to the findings of the scoping exercise, there is only enough time for 19 mappings, which includes 142 specific practices of the CMMI-Dev. The generic software development activities that are excluded from the study are the configurations management and risk management activities due to being more focused on management than software development activities.

After each CMMI-Dev specific goal is given a rating, a coverage percentage for each process area was calculated showing how successful the XP practice satisfied that process area. The calculation was based on the total quantity of goals in that process area and to what extent XP practices satisfied them. E.g. If three goals are satisfied, two are partially satisfied and one is unsatisfied, then a percentage coverage would show that the XP practice has a coverage of 50% being satisfied, 33% being partially satisfied and the remaining 17% unsatisfied for that process area. The percentage coverage for the trial mapping is shown in Figure 10.

Trial Coverage Percentage

Coverage Calculation	Requirements Development	40% Satisfied 30% Partially Satisfied 30% Unsatisfied	Planning Game

Figure 10: Trial Coverage Percentage

The results of the mappings and calculations thus satisfies the hypothesis set out earlier in the report by being able to provide evidence on relationships between XP practices and specific CMMI-Dev process areas.

The calculations and mappings between XP practices and CMMI-Dev process areas could then also reveal to organisations, certain practices of XP that may satisfy a specific process area of a current CMMI framework that the organisation may want to improve. Further research could then be built upon these mappings and results that could inspire future projects researching into blending quality management processes with XP practices to produce more efficient software development processes and project management hybrids.

4.0 Evaluation

The mapping between CMMI process areas and XP practices considers the staged representation of the CMMI-DEV model, version 1.2, released in August, 2006. This evaluation considers only the CMMI-Dev process areas and XP practices identified in the literature review as being relevant to this study and can be found in Figure 6, page 44. For each process area, a mapping between its specific practices to fulfill each goal and the XP practice was carried out. After that, a coverage rating for each practice was established considering the following criteria in Figure 8. After the rating phase, a coverage percentage of each process area was calculated to determine how applicable the XP practice(s) were during the experiment. The results were then grouped and a complete view of the CMMI process areas coverage by XP practices was generated. Each mapping and the results are discussed in the following sections under each software development activity.

4.1 Software Requirements & Specification

The mapping result tables continue throughout the evaluation to show how each process area mapped with the associated XP practice, with the full results shown in Appendix A. The structure of the each result table adopted a similar approach based on the profound and well established explanatory experimental recordings of Marcal (2007) and Stojanovic (2004). The table shows the process area and a specific goal with the mapping results of the total amount of specific practices that were either unsatisfied, partially satisfied or satisfied by the XP practice. The process area CMMI-Dev acronym and specific goal result number are combined to show which process area and specific goal is being analysed in the discussion after the table. The coverage calculation reveals how successful overall the XP practice was at satisfying the requirements of the process area.

4.1.1 Requirements Development & The Planning Game

Table 1: Requirements Development & The Planning Game Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices Mapping Results	XP Practice
RD1	**PA:** Requirements Development **SG1:**Develop Customer Requirements	2/2 Partially Satisfied	Planning Game
RD2	**PA:** Requirements Development **SG2:** Develop Product Requirements	3/3 Unsatisfied	Planning Game
RD3	**PA:** Requirements Development **SG3:** Analyse and Validate Requirements	4/5 Satisfied 1/5 Partially Satisfied	Planning Game
Coverage Calculation	**Requirements Development**	**40% Satisfied** **30% Partially Satisfied** **30% Unsatisfied**	**Planning Game**

Satisfied Mapping Results

RD3 From the results in table 1, it shows that the planning game practice has a high satisfaction rating in specific goal three, analyse and validate requirements. This is because the requirements development process area requires engagement with relevant stakeholders using methods for eliciting needs, expectations, constraints, and external interfaces. The planning game practice uses requirement gathering techniques like use cases and storyboards to show how the system will interact with components, other systems and users, which contributes to the high satisfaction rating.

The planning game practice again links well with the analysis and validation requirements goal as the development team performs feasibility studies and prioritisation of the requirements thus satisfying the need to prepare requirements before proceeding with development. Another key point that satisfies specific goal three is that the client is involved throughout the development. After each iteration the client is used to validate the working software and client feedback can then be taken and integrated into the next iteration.

Unsatisfied Mapping Results

RD2 The planning game practice was completely unsuccessful when mapping the XP planning game practice to specific goal two, developing product requirements. The product requirements are the expression of the client requirements in technical terms that can be used for design decisions and other than client requirements, this goal sets out to understand business and process requirements. The planning game practice does not satisfy this goal as XP produces an agile design that is ever evolving as changes are continuously made. Thus technical terms are not established as they would change per iteration due to the design evolving.

RD1 The results in table 1 show an overall low satisfaction mapping between the XP planning game and CMMI-Dev requirements development process area. From the results, the main contribution to this is that the planning game practice focuses profoundly on the customer, developing around their requirements and allocates minimal resources on analysing the business and process requirements involved in the project. This can be expected however, due to XP's focus on individuals and interactions over processes and tools as emphasised by the Agile Manifesto (Kent Beck et al. 2001) of which XP is partly based upon.

Further Analysis

When reflecting on the analysis and validation requirements goal and the XP planning game practice, it should be expected that both relate. The aim of both is to justify and validate why that requirement is needed before continuing on to design and implementation. The planning game practice does this in an early key point by performing feasibility studies and prioritising requirements resulting in a high

mapping rating to specific goal three. Although system requirements might evolve dramatically over time, XP integrates feedback on customer expectations and needs by emphasising short release cycles and continual customer involvement.

To increase the mapping rating between the planning game practice and the requirements development process area, it would be useful for the XP practice to have a pre-activity of analysing the product requirements before developing the client requirements. This would allow the development team to understand current processes and organisational workings instead of solely focusing on the client requirements.

4.1.2 Requirements Management & The Planning Game

Table 2: Requirements Management & The Planning Game Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices Mapping Results	XP Practice
REQM4	**PA:** Requirements Management **SG1:** Manage Requirements	1/5 Satisfied 2/5 Partially Satisfied 2/5 Unsatisfied	Planning Game
Coverage Calculation	**Requirements Management**	**20% Satisfied** **40% Partially Satisfied** **40% Unsatisfied**	**Planning Game**

Satisfied Mapping Results

REQM4 The results in table 2 show a low mapping score between requirements management and the planning game practice. The only specific practice to be fully satisfied out of the five specific practices was the obtain commitment to requirements specific practice. This specific practice deals with agreements and commitments among those who have to carry out the activities to implement the requirements. The planning game practice satisfied this specific practice as the team sets out an initial schedule / plan before each iteration commences, which will contain who has what tasks to complete. By documenting commitments in the schedule / plan before starting tasks, the planning game has a similar structure of documenting who performs what task thus satisfying the obtaining commitment to requirements.

Unsatisfied Mapping Results

REQM4 The managing of requirements is unsuccessfully handled by the planning game practice and is reflected in the table 2 results produced by the experiment. The requirements management process area requires changes to requirements to be documented and evaluated to determine how they affect the project. To an extent the planning game practices does this by setting up a review with the client, after each

iteration, to determine if changes need to be made although no criteria or process is setup to document and record the changes. If changes to requirements were recorded, this would allow traceability throughout the project to monitor the affect of the changes and identify if they were to have a negative or positive affect on the project.

The planning game practice fails to track requirements from beginning to the end of production, which the requirements management goal supports. The client or developer will often make decisions that are not documented in the 2-4 week XP development cycle thus cannot be traced through development to understand why they were made or what impact they have had on production.

There are also no procedures in the planning game practice that address inconsistencies between requirements that may result in problems between requirements later in the project. This is to be expected by the XP practice as it focuses more on encouraging development, which discovers requirements, unlike the CMMI-Dev process area that analyses and documents all requirements upfront before the design and development stages commence.

Further Analysis

To improve the mapping between the requirements management process area and the planning game practice, it would be beneficial for the planning game practice to include a requirements management process or structure. This would deal with monitoring and recording requirements and would be beneficial to an organisation to be able to trace requirements through the project to check if they are being satisfied.

XP performs particular activities to manage requirements without explicitly having a process in place to do it. The activities could be adapted in order to fully satisfy the requirements management process area. XP integrates requirements by using feedback on customer expectations and needs by emphasising short release cycles and continual customer involvement thus when testing is performed, the requirement is verified and validated to check that it fulfils the client needs.

4.2 Design

4.2.1 Technical Solution & Design Improvement

Table 3: Technical Solution & Design Improvement Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices Mapping Results	XP Practice
TS5	**PA:** Technical Solution **SG1:** Select Product Component Solutions	2/2 Unsatisfied	Design Improvement
TS6	**PA:** Technical Solution **SG2:** Develop The Design	1/4 Partially Satisfied 3/4 Unsatisfied	Design Improvement
TS7	**PA:** Technical Solution **SG3:** Implement The Product Design	1/2 Partially Satisfied 1/2 Unsatisfied	Design Improvement
Coverage Calculation	**Technical Solution**	**25% Partially Satisfied** **75% Unsatisfied**	**Design Improvement**

Satisfied Mapping Results

TS6 The results in table 3 show that the design improvement practice rates lowly in mapping with the technical solution process area. There are only two specific practices that are partially satisfied by the design improvement practice and they are in specific goal two and specific goal three. This is due to an initial design being mapped out by the design improvement practice but not to the extent that the specific practice requires such as architectural designs, design rules, and design modelling. The design improvement practice is more focused on developing whilst improving the designing as the project progresses, rather than producing supporting documentation thus contributing to the low success rating.

TS7 The design improvement practice uses refactoring techniques to remove unnecessary coding which can be applied to specific goal three of the technical solution as it focuses on implementing the design. Techniques used by the design improvement practices, such as refactoring, could be used by specific goal three to produce an improved design as the developers change from the design phase to implementation.

Unsatisfied Mapping Results

TS5 The technical solution process area requires alternative design solutions to be identified and analysed to enable the selection of a balanced solution across the life of the project in terms of cost, schedule, and performance. The design improvement

practice does not satisfy this as the design is continually evolving throughout the project thus there is no need to determine alternative solutions to the project.

TS7 The focus on producing documents such as user manuals, online help guides, maintenance and operator manuals by specific goal three contributes to a low mapping rating. The design improvement practice is only concerned with the actual design of the product and not the supporting documentation.

Further Analysis

The low rating between the technical solution and design improvement practice can be expected as the design improvement focuses on techniques like refactoring that can refine and improve the design as coding progresses, where as the technical solution focuses on having all the design specified before developing commences. This is based upon a clash of paradigms where the CMMI-Dev follows a more traditional waterfall approach and XP an incremental methodology of development. Specific goal one could be more fuller satisfied if the design improvement practice were to produce alternative designs. This is highly unlikely though as the design in XP evolves as the project progresses and would also take considerable time away from developing software compromising the quick delivery time of XP to produce software fast and frequently.

To increase the mapping rating between the design improvement practice and the technical solution process area, it would be useful to have software tools that reverse engineer the product designs. This could produce the documentation that is required by specific goal three and further map the design improvement practice and technical solution process area.

4.2.2 Technical Solution & Simple Design

Table 4: Technical Solution & Simple Design Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices Mapping Results	XP Practice
TS8	**PA:** Technical Solution **SG1:** Select Product Component Solutions	2/2 Unsatisfied	Simple Design
TS9	**PA:** Technical Solution **SG2:** Develop The Design	1/4 Partially Satisfied 3/4 Unsatisfied	Simple Design
TS10	**PA:** Technical Solution **SG3:** Implement The Product Design	1/2 Partially Satisfied 1/2 Unsatisfied	Simple Design
Coverage Calculation	**Technical Solution**	**25% Partially Satisfied 75% Unsatisfied**	**Simple Design**

Satisfied Mapping Results

TS9 The results show in table 4, that there are no specific practices of the technical solution process area that are fully satisfied by the simple design practice. The specific practice of designing the product, in specific goal two, is partially satisfied by the simple design practice as an initial design is mapped out but not to the extent that the specific practice requires. The specific practice requests architectural designs, design rules, design modelling, data schemas, algorithms to be developed, and heuristics to be applied, where as the simple design practice does not provide that amount of design detail. It is more concerned with how to design, rather than producing large amounts of supporting documentation.

TS10 The simple design practice suggests techniques to remove unnecessary coding which partially satisfies the specific practice, implementing the design, in specific goal three. Although this specific practice focuses more on implementation, techniques such as refactoring can be used during the phase from design to implementation to improve the design by keeping its structure simple.

Unsatisfied Mapping Results

TS8 The technical solution process area has a coverage rating of being 75% unsatisfied by the simple design practice. Specific goal one, select product component solutions, requires alternative solutions and selection criteria to be established which acts as a backup to any chosen solution that does not fulfill its potential. The simple design practice does not satisfy this specific goal as it emphasises keeping the design uncomplicated. As change in a project is inevitable, the XP practice focuses on producing a flexible design in order to manage change more successfully as the project progresses.

TS9 Specific goal two, developing the design, requires a technical data package to be established. This provides the developer with a comprehensive description of the product as it is developed. The simple design practice does not establish a technical data package as the amount of documenting and recording of information about the product, over the design phase, requires a large period of time. The simple design practice does not satisfy specific goal two as it focuses on improving the design, instead of documenting information about the software products.

TS10 Specific goal three, implementing the product design, results in being unsatisfied by the simple design practice. When implementing the design, the simple design practice can suggests techniques to remove unnecessary coding which can be applied to this specific goal. Although this specific goal focuses more on implementation, techniques such as refactoring can be used during the phase from design to implementation to improve the design. The specific goal requires the development of product support documentation, which is unsatisfied by the simple design practice. Documents produced by the specific goal are user manuals, online help guides, maintenance and operator manuals where as the simple design practice is only concerned with the actual design of the product and not the supporting documentation.

Further Analysis

The technical solution process area does not map successfully to the simple design practice as the simple design practice designs only for the specified functionality and does not include anything that will not be of use. The design process is continuous throughout the life of the project in XP where as the technical solution focuses on a large amount of upfront design before implementation commences. To improve the mapping between the technical solution process area and the simple design practice, techniques like refactoring could be transferred to the technical solution process area and used to focus more on design during implementation. This would make the design more flexible to changes as they occur in the implementation phase, which could possibly reduce re-working and maintenance costs for an organisation.

4.3 Implementation

4.3.1 Product Integration & Pair Programming

Table 5: Product Integration & Pair Programming Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices Mapping Results	XP Practice
PI11	**PA:** Product Integration **SG1:** Prepare for Product Integration	3/3 Unsatisfied	Pair Programming
PI12	**PA:** Product Integration **SG2:** Ensure Interface Compatibility	2/2 Unsatisfied	Pair Programming
PI13	**PA:** Product Integration **SG3:** Assemble Product Components and Deliver the Product	1/4 Satisfied 3/4 Unsatisfied	Pair Programming **Further Satisfy Goal** Planning Game
Coverage Calculation	**Product Integration**	**11% Satisfied** **89% Unsatisfied**	**Pair Programming**

Satisfied Mapping Results

PI13 The pair programming practice overall has a very low mapping rating with the product integration process area as shown in table 5. The only satisfied specific practice is in specific goal two where the pair programming practice is achieving the same goal by building the components of the system, the physical coding of the components.

Unsatisfied Mapping Results

PI11 Product Integration establishes and maintains the environment needed to support the integration of the product components. This could be the decision to buy an off the shelf development environment or develop, as new, an integrated development environment to build the software. The pair programming practice does not consider the development environment as the main focus is solely on developing the software.

The product integration requires procedures and criteria to be established that define how the product components are to be verified, functions they are expected to have, and when the product is fully assembled, how it should be validated and delivered. The pair programming practice does not satisfy this aspect of the process area as it

does not establish procedures and criteria, only techniques that can be used to improve the quality of the code.

PI12 Specific goal two is fully unsatisfied by the pair programming practice. The pair programming practice does not satisfy the review interface descriptions specific practice as it deals with managing interface requirements, specifications, and designs to help ensure that implemented interfaces will be complete and compatible. The second specific practice that is unsatisfied in specific goal two, manages interfaces including maintenance of the consistency of the interfaces throughout the life of the system. The pair programming practice does not satisfy this specific practice as it is concerned with managing the interfaces where as the pair programming practice relates to physical coding and problem solving.

PI13 Specific goal three has three out of four specific practices unsatisfied. Specific goal three requires an evaluation to be performed that involves examining and testing assembled product components for performance, suitability and readiness using the available procedures and environment that the organisation have established. Specific goal three does not relate fully to implementation, the focus is more on testing thus the pair programming practice does not satisfy this specific practice. The planning game practice would however further satisfy this specific goal as client evaluations are performed after each iteration that would satisfy the performance, suitability and readiness criteria.

Further Analysis

To have the product integration process area and pair programming practice not map successfully was not expected as within the experiment framework, they are the only implementation activities. A hybrid of both would be more successful with the product integration process area specifying what needs to be completed to integrate the system and the pair programming practice providing how the implementation should be performed.

The XP pair programming activity performs product integration activities but there is no set structure or process to monitor the progress. The pair programming practice has the ability to further satisfy this process area, it just needs to be adapted into a structured process in order to map more precisely to the CMMI-Dev.

4.4 Validation & Verification

4.4.1 Validation & Client Tests

Table 6: Validation & Client Tests Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices Mapping Results	XP Practice
VAL14	**PA:** Validation **SG1:** Prepare for Validation	2/3 Satisfied 1/3 Partially Satisfied	Client Tests
VAL15	**PA:** Validation **SG2:** Validate Product or Product Components	2/2 Satisfied	Client Tests
Coverage Calculation	**Validation**	**80% Satisfied** **20% Partially Satisfied**	**Client Tests**

Satisfied Mapping Results

VAL14 The client tests practice has a high mapping rating with the validation process area as shown in table 6. This is due to products and product components being selected for validation on the basis of their relationship to the client needs. The client tests practice satisfies this as the needs of the client are being tested due to the client themselves preparing the tests including acceptance testing.

VAL15 When performing the validation, the results are obtained where they can be analysed to check whether they achieve the client's needs. The client tests practice map well to performing the validation as they involve the client and the documentation produced is common to both such as validation reports and results, run logs and client demonstrations. The client tests practice satisfies specific goal two as the results are analysed to reveal if the client's needs are met. As the tests involve the client, feedback is given that reports whether the tests are successful or fail, and will reveal a probable cause of malfunction. The client tests satisfy this goal as the results are collected in both practices and determine whether to proceed with the project or to correct any problems that the tests have detected.

Unsatisfied Mapping Results

VAL14 The specific practice that is not fully satisfied by the client tests practice is within specific goal one, which involves establishing validation procedures and criteria. Validation procedures and criteria are defined to ensure that the product or product component will fulfill its intended use when placed in its intended environment. The client tests practice partially satisfies this specific practice as client acceptance test

cases meet the need of validation procedures. Documentation such as performance thresholds, standards and environmental performance will not be produced by the client tests thus the specific practice is only partially satisfied.

Further Analysis

The successful mapping between the client tests practice and validation process area was expected. By involving the client in the tests, their expectations are more fully understood by the development team. The client tests satisfy the validation specific practices as the requirements of the system are checked that they conform to the requirements specification. An organisation that was using XP and were introducing the CMMI-Dev framework, could analyse the results produced by this experiment and know that this process area is 80% satisfied by the client tests practice.

4.4.2 Validation & Small Releases

Table 7: Validation & Small Releases Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices Mapping Results	XP Practice
VAL16	**PA:** Validation **SG1:** Prepare for Validation	2/3 Satisfied 1/3 Unsatisfied	Small Releases
VAL17	**PA:** Validation **SG2:** Validate Product or Product Components	2/2 Satisfied	Small Releases
Coverage Calculation	**Validation**	**80% Satisfied** **20% Unsatisfied**	**Small Releases**

Satisfied Mapping Results

VAL16 Table 7 shows results of the small releases practice mapping extremely well with the validation process area with only one specific practice being unsatisfied throughout. The small release practice satisfies the selection of products and product components by emphasising early testing that focuses on producing working software frequently. The products and product components can be tested with the client and used to determine whether the software satisfies their needs.

VAL17 The small release practice satisfies specific goal two fully by performing validation tests with the client. Results are obtained after the testing of a working piece of software is produced where they can be analysed to check whether they achieve the client's needs. The small release practice satisfies this specific practice as the results of the tests are analysed to reveal if the client's needs are met. As the tests in the small release involve the client, feedback is given that reports whether the tests are successful. The client tests involved in each small release satisfy this specific practice as the results are collected in both practices and determine whether to proceed with

building further requirements on the next iteration or to correct any problems that the tests have detected.

Unsatisfied Mapping Results

VAL16 The validation process area requires procedures and criteria to be defined to ensure that the product or product component will fulfill its intended use when placed in its intended environment. The small release practice does not satisfy this aspect of the first specific goal as it focuses on frequent releases and building tested software in small stages. The validation process area requires a testing criteria and procedures to be documented including performance thresholds, standards and environmental performance, which is not produced by the small releases practice.

Further Analysis

The mapping between the validation process area and small release practice was expected to have a high rating. This was due to building in small increments where the software produced during that increment could be tested to check if it conforms to the client's requirements. The analysis from the testing results could then be taken and incorporated into the next iteration to improve or change functionality depending on what the client requests. An organisation that was using the CMMI-Dev framework and wanted to integrate an agile development practice like XP, could analyse the results produced by this experiment and know that their process area of validation is 80% satisfied by the small releases practice.

4.4.3 Validation & Pair Programming

Table 8: Validation & Pair Programming Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices Mapping Results	XP Practice
VAL18	**PA:** Validation **SG1:** Prepare for Validation	1/3 Partially Satisfied 2/3 Unsatisfied	Pair Programming
VAL19	**PA:** Validation **SG2:** Validate Product or Product Components	1/2 Partially Satisfied 1/2 Unsatisfied	Pair Programming
Coverage Calculation	**Validation**	**40% Partially Satisfied 60% Unsatisfied**	**Pair Programming**

Successful Mapping Results

VAL18 The validation process area and pair programming practice unsuccessfully maps with no specific practice being fully satisfied as shown in table 8. When selecting the products for validation based on their relationship with the client's needs, the client and the developers who work on that specific piece of functionality select what

should be chosen and validated. This is the most logical method of choosing products to validate as the developers are the most suitable in the team to make these decisions as it is their own developed software that is being validated.

VAL19 The pair programming practice only partially satisfies performing the validation in specific goal two. This is due to having the two initial developers programming together throughout the implementation phase. As one developer codes, the other analyses his partners code to check that it conforms to the specifications.

Unsuccessful Mapping Results

VAL18

VAL19 The pair programming practice does not satisfy establishing a validation environment, validation procedures, or criteria as required by specific goal one as no testing is performed during pair programming as it focuses only on coding of the design. No validation results can be analysed, as requested by specific goal two, as the pair programming practice does not error detect or produce documentation thus remains unsatisfied.

Further Analysis

The unsuccessful mapping of the validation process area and the pair programming practice was to be expected due to the pair programming practice being a technique to implement the design and the validation process area focusing on the product trying to fulfil its intended use. The pair programming practice and validation process area contains activities that do not successfully map.

4.4.4 Validation & Test Driven Development

Table 9: Validation & Test Driven Development Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices Mapping Results	XP Practice
VAL20	**PA:** Validation **SG1:** Prepare for Validation	2/3 Satisfied 1/3 Unsatisfied	Test Driven Development
VAL21	**PA:** Validation **SG2:** Validate Product or Product Components	2/2 Satisfied	Test Driven Development
Coverage Calculation	**Validation**	**80% Satisfied** **20% Unsatisfied**	**Test Driven Development**

Satisfied Mapping Results

VAL20 The validation process area and the test driven development practice map successfully with an 80% coverage rating as shown in table 9. The test driven development satisfies the selecting products for validation specific practice by focusing on the clients' needs from the beginning of development as the developers are forced to test first. This allows products to be selected based on an understanding of the client requirements and needs.

VAL21 When performing the validation, the test driven practice successfully maps as the client's needs are focused upon during testing and then coded. Any requirements that are unclear can be clarified by the client to allow the developer to gain a thorough understanding of their needs. This tests that the product or piece of functionality fulfills its intended use.

The test driven development practice produces test results that can be analysed by the development team and the client which satisfies specific goal two. The test driven development practice produces results that act as a safety precaution to the developer as they know that a change to the code has passed or failed before continuing with development.

Unsatisfied Mapping Results

VAL20 The test driven development practice does not setup validation procedures or criteria that are required by specific goal one. The test driven development practice does not satisfy this as it focuses on testing and coding rather than establishing procedures and criteria.

Further Analysis

The validation process area and test driven development was expected to produce a high mapping rating. This was due to the continuous testing that the test driven development practice enforces throughout the project, which helps to satisfy the validation aim of proving that a piece of code or functionality fulfills its intended use. An organisation using XP, who wanted CMMI-Dev certification, could view this result and know that test driven development maps to the CMMI-Dev validation process area strongly.

4.4.5 Verification & Client Tests

Table 10: Verification & Client Tests Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices (SP) Mapping Results	XP Practice
VER22	**PA:** Verification **SG1:** Prepare for Verification	2/3 Satisfied 1/3 Unsatisfied	Client Tests
VER23	**PA:** Verification **SG2:** Perform Peer Reviews	3/3 Unsatisfied	Client Tests **Further Satisfy Goal** Planning Game + Pair Programming
VER24	**PA:** Verification **SG3:** Verify Selected Work Products	2/2 Satisfied	Client Tests
Coverage Calculation	**Verification**	**50% Satisfied** **50% Unsatisfied**	**Client Tests**

Satisfied Mapping Results

VER22 Table 10 shows that the mapping between the verification process area and the client tests practice produces a 50% successful mapping. When selecting the products to be verified, the client test practice satisfies this by focusing on the clients' requirements from the beginning of development by being forced to test first. This allows work products to be selected based on an understanding of the client, their requirements and their needs. Methods that are common to both specific practice and the client tests practice are test cases and acceptance tests.

VER24 When performing the verification, in specific goal three, the client tests practice satisfies this specific practice as verification results from client tests are obtained where they can be analysed to check whether they achieve the project requirements and objectives. Documentation produced that is common to both is validation reports and results, run logs and client demonstrations. As the tests involve the client, feedback is given that reports whether the tests are successful, or will reveal a probable cause of failure. The client tests satisfy the specific practice of analysing the results, where in both, the results are collected and determine whether to proceed with development on the next iteration or to correct any problems that the tests have detected.

Unsatisfied Mapping Results

VER22
VER23 The client tests practice and the verification process area have a 50% unsuccessful mapping as shown in table 10. The verification process area requires procedures and criteria to be established in order to verify products which the client tests practice

does not satisfy. The verification process area also requires peer reviews to be performed that examine the work and identify defects for removal and recommend other changes to the work products. Peer reviews are not conducted in the client tests practice thus specific goal two remains fully unsatisfied.

Further Analysis

It was expected that the client tests practice would satisfy the verification process area as the client tests focuses on justifying to the developers and client that the product is operating as expected. The tests produce data that is analysed for defects, which the verification process requires. To further map the client tests practice to the verification process area, key points of the planning game practice and pair programming practice could be added to satisfy specific goal two, peer reviews. The planning game practice incorporates evaluations and reviews with stakeholders where as the pair programming practice physically codes in twos. This would allow the second programmer to objectively analyse and review the code produced which would further map specific goal two in this mapping.

4.4.6 Verification & Small Releases

Table 11: Verification & Small Releases Mapping Results

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices (SP) Mapping Results	XP Practice
VER25	**PA:** Verification **SG1:** Prepare for Verification	2/3 Satisfied 1/3 Unsatisfied	Small Releases
VER26	**PA:** Verification **SG2:** Perform Peer Reviews	3/3 Unsatisfied	Small Releases **Further Satisfy Goal** Planning Game + Pair Programming
VER27	**PA:** Verification **SG3:** Verify Selected Work Products	2/2 Satisfied	Small Releases
Coverage Calculation	**Verification**	**50% Satisfied** **50% Unsatisfied**	**Small Releases**

Satisfied Mapping Results

VER25 In table 11, the results show that 50% of the verification process area can be successfully mapped with the small releases practice. Specific goal one, prepare for verification, requires the selection of work products to be verified. The small releases practice satisfies this due to its focus on early testing of working software. When a

piece of software is released, it is selected to be tested based on the understanding of the client and their requirements. The verification environment is established by the small release practice, as without having an environment to test the small releases, the small release practice would not be effective.

VER27 Specific goal three, the verification of the work product, is fully satisfied by the small release practice. The verification is performed by testing the small releases of working software where results can be obtained and analysed to verify whether they achieve the client's needs. As the tests of the small release practice involve the client, feedback is given that reports whether the software is what they require. Both the small release practice and verification process area are able to analyse the results and determine whether to proceed with the project or to correct defects that have been discovered during the verification activities.

Unsatisfied Mapping Results

VER25 The small release practice has a 50% unsuccessful mapping with the verification process area. Specific goal one, preparing for verification, requires that verification procedures and criteria be established to ensure that the work products meet their requirements. The small release practice does not satisfy this aspect of preparing for verification as it focuses on releasing software frequently and in increments. To further satisfy the mapping, the small release practice would need to produce documents that include set standards, supplier agreements, procedures and policies.

VER26 Specific goal two, perform peer reviews, requires reviews that examine the work produced, identifies defects for removal and records recommendations for any other changes to the work. Peer reviews are not conducted in the small release practice thus specific goal two remains fully unsatisfied.

Further Analysis

The small release practice was expected to map to the verification process area to an extent due to the emphasis on testing. By releasing in small phases, the piece of software can be tested to verify that it meets its specification. To further map the small releases practice to the verification process area, key points of the planning game practice and pair programming practice could be added to satisfy specific goal two, peer reviews. The planning game practice incorporates evaluations and reviews with stakeholders where as the pair programming practice physically codes in twos. This would allow the second programmer to objectively analyse and review the code produced which would further map specific goal two in this mapping. An organisation who currently use XP, and want to have a more structured development framework to producing software, could anaylse these results to show that their current planning game and small release practice may satisfy the CMMI-Dev verification process area.

4.4.6 Verification & Pair Programming

Table 12: Verification & Pair Programming Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices (SP) Mapping Results	XP Practice
VER28	**PA:** Verification **SG1:** Prepare for Verification	1/3 Partially Satisfied 2/3 Unsatisfied	Pair Programming
VER29	**PA:** Verification **SG2:** Perform Peer Reviews	1/3 Satisfied 2/3 Unsatisfied	Pair Programming **Further Satisfy Goal** Planning Game
VER30	**PA:** Verification **SG3:** Verify Selected Work Products	1/2 Partially Satisfied 1/2 Unsatisfied	Pair Programming
Coverage Calculation	**Verification**	**20% Satisfied** **20% Partially Satisfied** **60% Unsatisfied**	**Pair Programming**

Satisfied Mapping Results

VER28
VER30 The pair programming practice and the verification process area has a low 20% coverage rating which is shown in table 12. When selecting the products for verification, the developers in the pair programming practice select what should be tested and are the most suitable on the team to make that decision as it is their code that is being verified. When performing the validation in specific goal three, the pair programming practice partially satisfies it as some validation is performed continuously by working in pairs. By programming in pairs, any coding that is performed has at least been checked by the two initial developers. Throughout the implementation, as one codes the other analyses his partners code for errors, which acts as a minimal form of validation.

Unsatisfied Mapping Results

VER28 The results show an 80% unsatisfactory coverage rating between the verification process area and the pair programming practice. The pair programming practice has a low mapping with specific goal one, preparing for verification. The pair programming practice does not establish a verification environment as no testing is performed during pair programming and no procedures or criteria are created to ensure that the software meets requirements. The pair programming practice focuses on implementing the design and less on verification of requirements.

VER29 Specific goal two, perform peer reviews, requires reviews that examine the work produced, identifies defects for removal and records recommendations for any other changes to the work. Preparation for peer reviews is not conducted in the pair programming practice as the reviews are a continuous process with one developer methodically examining the work of the other. The pair programming practice does not satisfy the third specific practice of goal two as the data is not stored for future reference or analysed to discover why the error occurred.

Further Analysis

Testing contributes to verification greatly where as the pair programming practice focusing more on implementing the design than testing although by having developers code together, they can peer review one another, which can act as a minimal form of verification. To further increase the mapping between the verification process area and the pair programming practice, the planning game practice could be added to improve the mapping between specific goal two where the evaluations and meetings setup by the planning game may be used as peer reviews throughout development. An organisation using the CMMI-Dev, may use these results to conclude that the pair programming practice would not improve their current process area of verification.

4.4.7 Verification & Test Driven Development

Table 13: Verification & Test Driven Development Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices (SP) Mapping Results	XP Practice
VER31	**PA:** Verification **SG1:** Prepare for Verification	2/3 Satisfied 1/3 Unsatisfied	Test Driven Development
VER32	**PA:** Verification **SG2:** Perform Peer Reviews	3/3 Unsatisfied	Test Driven Development **Further Satisfy Goal** Planning Game + Pair Programming
VER33	**PA:** Verification **SG3:** Verify Selected Work Products	2/2 Satisfied	Test Driven Development
Coverage Calculation	**Verification**	**50% Satisfied** **50% Unsatisfied**	**Test Driven Development**

Satisfied Mapping Results

VER31 Work products are selected based on their contribution to meeting project objectives and project requirements which reflects the results shown in table 13. The test driven development satisfies this specific goal by focusing on the clients' needs from the beginning of development as they are forced to test first. This allows products to be selected based on an understanding of the client requirements and needs as required by specific practice one, select products for verification. The test driven development practice satisfies specific practice two, establish the verification environment as testing is the main objective of the XP practice. Without the testing environment the practice would be useless thus an established verification environment is established.

VER33 Verification is performed throughout the test driven practice as the client's needs are focused upon during testing and then coded. Any requirements that are unclear can be clarified by the client to allow the developer to gain a thorough understanding of the client's needs. This allows the product or piece of functionality to demonstrate that it fulfills the project requirements and objectives. To satisfy this specific goal, the test driven development practice records results, produces run logs and reports to be analysed.

Unsatisfied Mapping Results

VER31 When preparing for verification, the test driven development practice does not establish procedures or criteria to follow during verification as its main focus is on testing.

VER32 Specific goal two, perform peer reviews, requires reviews that examine the work produced, identifies defects for removal and records recommendations for any other changes to the work. Peer reviews are not conducted by the test driven development practice thus specific goal two remains fully unsatisfied.

Further Analysis

To further map the verification process area and the test driven practice, key points of the planning game practice and pair programming practice could be added to satisfy specific goal two, peer reviews. The planning game practice incorporates evaluations and reviews with stakeholders where as the pair programming practice physically codes in twos. This would allow the second programmer to objectively analyse and review the code produced which would further map specific goal two in this mapping.

4.5 Software Project Management

4.5.1 Casual Analysis and Resolution & The Planning Game

Table 14: Casual Analysis and Resolution & The Planning Game Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices (SP) Mapping Results	XP Practice
CAR34	**PA:** Casual Analysis & Resolution **SG1:** Determine Causes of Defects	1/2 Satisfied 1/2 Partially Satisfied	The Planning Game
CAR35	**PA:** Casual Analysis & Resolution **SG2:** Address Causes Of Defects	2/3 Partially Satisfied 1/3 Unsatisfied	The Planning Game
Coverage Calculation	**Casual Analysis & Resolution**	**20% Satisfied** **60% Partially Satisfied** **20% Unsatisfied**	**The Planning Game**

Satisfied Mapping Results

CAR34 Table 14 shows illustrates that the planning game practice has a low 20% mapping rating with the casual analysis and resolution process area. The planning game practice satisfies specific practice one, which focuses on selecting defects for analysis. The planning game practice does this by having the development team and client come together, after each 2-4 week iteration, and discuss any problems found during the iteration. The defects are highlighted and come from requirements analysis, testing or even discovered by the client. At these meetings, defects can be highlighted to determine the cost, time and resources needed to fix them and consideration can be given to how they may impact the project, which partially satisfies the second specific practice of goal one, analyse causes. As the whole team are present at these planning game meetings, those who have an understanding of the defect are responsible for performing the task to resolve the problem and plans to do this can be arranged at the meeting.

Unsatisfied Mapping Results

CAR35 The casual analysis and resolution process area has a 20% unsatisfied mapping rating by the planning game practice as shown in table 14. The specific practice of recording data in specific goal two is the only fully unsatisfied specific practice when mapping the planning game practice to the casual analysis and resolution process area. Data on the defect is recorded by the specific practice and the information would be used across the organisation on similar projects or to be stored and used by future projects

to learn from previous project errors. The planning game practice does not record defect data or solution data for use in future projects.

Further Analysis

CAR34 The results show that there are a remaining 60% of specific practices that are partially satisfied by the planning game practice in this process area. If the planning game practice were to be adapted more to suit the casual analysis and resolution process area, more partially satisfied practices would be fully satisfied. To more fully satisfy these specific practices, the planning game practice needs to produce supporting documentation when addressing the causes of the defects and evaluating the effect of the changes. Documents such as action proposals would allow an organisation to track why the defect occurred, monitoring the effect of resolving the defect and could be used to trace the defect through development to apply lessons learnt in future projects. Tools that could automatically generate defect data would also further improve the mapping between the planning game practice and the casual analysis and resolution process area.

4.5.2 Integrated Project Management & The Planning Game

Table 15: Integrated Project Management & The Planning Game Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices (SP) Mapping Results	XP Practice
IPM36	**PA:** Integrated Project Management **SG1:** Use the Project's Defined Process	2/6 Satisfied 2/6 Partially Satisfied 2/6 Unsatisfied	The Planning Game
IPM37	**PA:** Integrated Project Management **SG2:** Coordinate and Collaborate with Relevant Stakeholders	2/3 Satisfied 1/3 Partially Satisfied	The Planning Game
Coverage Calculation	**Integrated Project Management**	**45% Satisfied** **33% Partially Satisfied** **22% Unsatisfied**	**The Planning Game**

Satisfied Mapping Results

IPM36 The planning game practice has a 45% mapping rating with the integrated project management process area as shown in table 15. The planning game practice sets out activities that occur throughout the project's lifecycle and help staff and stakeholders establish the project's defined process satisfying specific practice one and two of specific goal one. Although organisational process assets differ from organisation to organisation, the planning game practice does set out a schedule for completion of activities during the project regardless of process assets.

IPM37 Specific goal two of the integrated project management process area deals with managing and coordinating stakeholder involvement. The planning game practice satisfies this by having scheduled stakeholder meetings after each iteration where any issues regarding the stakeholder can be raised and discussed. Part of specific goal two is to resolve any stakeholder coordination issues, which is fully satisfied by the planning game practice by having evaluation reviews after each iteration of development. All stakeholders are involved at this stage where coordination issues would be resolved.

Unsuccessful Mapping Results

IPM36 Table 15 shows that the planning game practice does not map with 22% of the integrated project management process area. The specific practice of integrating plans, in specific goal one, is fully unsatisfied due to it integrating the project with other plans that may affect the project such as software quality assurance plans, configuration management plans etc. The planning game practice sets out activities to be completed during the project and does not specify documents to be produced during the activities thus cannot satisfy the integrating project plans specific practice.

The planning game practice does not record processes or information that could be used in future projects. Documentation produced by specific goal one regards improvements that could be made to the project, processes and product measures collected and information that could be used in future projects. The planning game practice does not record information for future projects.

Further Analysis

The low mapping result can be expected between the integrated project management process area and the planning game practice due to the lack of overall project management in XP. The documentation required to be produced to support the specific practices fails in the mapping and is one of the main reasons why 33% of the specific practices remain partially satisfied. XP emphasises working software over documentation where as the integrated project management process area produces a large amount of supporting documentation that is not produced during the planning game practice, which justifies the unsatisfied and partially satisfied mapping results.

4.5.3 Measurement and Analysis & The Planning Game

Table 16: Measurement and Analysis & The Planning Game

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices (SP) Mapping Results	XP Practice
MA38	**PA:** Measurement & Analysis **SG1:** Align Measurement & Analysis Activities	1/4 Partially Satisfied 3/4 Unsatisfied	The Planning Game
MA39	**PA:** Measurement & Analysis **SG2:** Provide Measurement Results	4/4 Unsatisfied	The Planning Game
Coverage Calculation	**Measurement & Analysis**	**12.5% Partially Satisfied** **87.5% Unsatisfied**	**The Planning Game**

Satisfied Mapping Results

MA38 The results in table 16 show that the measurement and analysis process area has no matching relationships with the planning game process. The only specific practice to be partially satisfied was to establish measurement objectives. The planning game practice partially satisfies this specific practice as it fulfills establishing measurement objectives by allowing the client and development team to plan objectives prior to coding. The specific practice requires reviews and documentation to be produced to enable traceability throughout the project thus the planning game practice only partially satisfies that specific practice by having frequent iteration reviews.

Unsuccessful Mapping Results

MA38 The results show that 87.5% of the measurement and analysis process area is unsatisfied by the planning game practice. Specific goal one, align measurement and analysis activities, is nearly fully unsatisfied due to the planning game practice not specifying measurements to achieve objectives set out earlier in development. The remaining specific practices are also unsatisfied and regard data collection and storage procedures where the planning game practice has no mechanisms setup to specify how to collect measurement data or to store data that could be used for further use.

MA39 Specific goal two, provide measurement results, is fully unsatisfied as no measurement results are collected, analysed, stored for future use or communicated to the development team. If there were measurement results to be discussed, then this could be done after each iteration at the evaluation meetings. This would allow action to be taken in the next iteration based on the results of the measurement data.

Further Analysis

XP is applied in a project to project basis, one project at a time, where as this CMMI-Dev framework would be used for every project in the same rigorous manner regardless of project variation. To increase the mapping of the measurement and analysis process area with the planning game practice, there would need to be more project management involved in the XP practices. This would allow monitoring and tracking of measurements during development and could the results could be reviewed after each iteration.

4.5.5 Project Monitoring and Control & The Planning Game

Table 17: Project Monitoring and Control & The Planning Game

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices (SP) Mapping Results	XP Practice
PMC40	**PA:** Project Monitoring & Control **SG1:** Monitor Project Against Plan	3/7 Satisfied 3/7 Partially Satisfied 1/7 Unsatisfied	The Planning Game **Further Satisfy Goal** Project Management
PMC41	**PA:** Project Monitoring & Control **SG2:** Manage Corrective Action to Closure	2/3 Satisfied 1/3 Partially Satisfied	The Planning Game
Coverage Calculation	**Project Monitoring & Control**	**50% Satisfied** **40% Partially Satisfied** **10% Unsatisfied**	**The Planning Game**

Satisfied Mapping Results

PMC40 The project monitoring and control has a 50% mapping coverage with the planning game practice as shown in table 17. Specific goal one requires progress and milestone reviews to be conducted, which the planning game practice incorporates into development. The reviews are performed before and after each iteration and can include issues on evaluation testing, project status, project measurements and development problems. The aim of the project monitoring and control process area is to allow the project to progress under supervision and allow any issues that may arise to be communicated to the team so everyone is aware of project status. The planning game practice maps successfully to these progress and milestone reviews as communication involving all stakeholders is persistent throughout an XP project. This is achieved by having meetings and reviews before and after each cycle that could involve progress and milestone issues.

PMC41 Specific goal two, manage corrective action to closure, is almost fully satisfied by having two out of three specific practices successfully mapped to the planning game practice. The evaluation reviews in the planning game practice, allow for issues with development to be analysed and corrective action plans to be put in place for the next iteration. This goal could be further satisfied if the planning game practice were to have a process that recorded the analysed data and stored what changes were to be made in the next iteration. This data could be used for future projects and would fully satisfy the final specific practice, managing the corrective action, of specific goal two.

Unsatisfied Mapping Results

PMC40 The results in table 17, show that there was only 10% of the project monitoring and control process area that was unsuccessfully mapped to the planning game practice. The specific practice fully unsatisfied by the planning game practice was the monitoring data management specific practice in specific goal one. This is due to the planning game practice not having any management structure to monitor project data that could be useful in determining whether the project is on schedule, within budget or remaining resources are suitable to complete the project.

Further Analysis

The results show that 40% of the project monitoring and control process area is partially satisfied by the planning game practice. To improve the mapping between the CMMI-Dev process area and the XP practice, the planning game practice requires more project management to allow project resources to be monitored as the project progresses. Within specific goal one, monitoring the project against the plan, the specific practices of monitoring project parameters, monitoring risks and monitoring stakeholder involvement are partially satisfied as the planning game practice uses evaluation reviews where these factors can be addressed. The reason they are not fully satisfied is due to the planning game practice not monitoring or documenting them as the development progresses. This would allow deviations from the plan to be dealt with at regular reviews and action to be taken in the next planning game iteration.

4.5.6 Project Planning & The Planning Game

Table 18: Project Planning & The Planning Game Mapping Result

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices (SP) Mapping Results	XP Practice
PP42	**PA:** Project Planning **SG1:** Establish Estimates	3/4 Partially Satisfied 1/4 Unsatisfied	The Planning Game **Further Satisfy Goal** Project Management
PP43	**PA:** Project Planning **SG2:** Develop a Project Plan	7/7 Unsatisfied	The Planning Game
PP44	**PA:** Project Planning **SG3:** Obtain Commitment to the Plan	3/3 Unsatisfied	The Planning Game
Coverage Calculation	**Project Planning**	**21% Partially Satisfied** **79% Unsatisfied**	**The Planning Game**

Satisfied Mapping Results

PP42 Table 18 shows that no specific practice of the project planning process area, fully maps with the planning game XP practice. Specific practices that are partially satisfied are estimating the scope of the project, establishing work products and task attributes, and determining estimates of effort and cost. To understand the scope of the project, the planning game practice involves the client throughout development to fully understand and discover their needs and requirements. Understanding the scope of the project could be more fully mapped by the planning game practice if the development team were to analyse the environment of the system instead of only relying on the client's requirements on which to build the system. This would allow the developers to understand current systems, reveal present methods of completing tasks and the opportunity to discover valuable requirements that the client may not reveal due to inexperience in requirements elicitation.

The developers estimate effort and costs for the requirements that have been identified but to fully satisfy the specific practice, it requires an estimation criteria and totals for project cost and effort to be established. The planning game practice does not specify any activities to establish a fully specified project plan which would include set resources such as budget and time. The lack of project management in the planning game practice is evident and results in a poor mapping with the project planning process area.

Unsatisfied Mapping Results

PP42 The coverage rating shows that 79% of the specific practices are unsatisfied by the planning game practice. Specific goal one, establish estimates, requires a project lifecycle to be defined, however, the planning game practice performs the same activities regardless of project. The planning game practice assumes that every project is suitable to apply the practices of XP resulting in an unsuccessful mapping as not all lifecycles can effectively adopt an agile development style.

PP43 Specific goal two, develop a project plan, is fully unsatisfied by the planning game practice. The planning game practice does not identify a budget for the project but does set out a schedule based on task estimates which partially satisfies specific practice one. This is extremely risky as there is no set processes to monitor resources setout in XP. The risks in the planning game practice are not identified prior to starting the project but are dealt with as they develop leaving specific practice two unsatisfied. The XP practice relies on short development cycles encouraging testing and evaluations to occur that would highlight any risks to the project. The planning game practice does not consider project resources such as materials, tools, machinery and staff as it assumes they are provided to complete the project, which further widens the gap between the planning game practice and specific goal two. The skills and experience of the staff are also not taken into consideration by the planning game practice and it assumes that staff should be able to complete the activities of XP.

PP44 The results for specific goal three, obtain commitment to the plan, shows that the planning game practice does not satisfy any of the specific practices. Specific goal three requires plans that may affect the project to be reviewed, resources to be adjusted dependant on plan changes and to obtain commitment from individuals to adhere to the plan. XP focuses on developing quickly and releasing working software frequently where as this specific goal requires reviewing all plans and making sure they are consistent with one another. The planning game practice focuses on development rather than producing large amounts of documentation thus specific goal three remains fully unsatisfied.

Further Analysis

The unsuccessful coverage rating between the planning game practice and project planning process area was to be expected as XP embraces change, which results in little planning at the beginning of the project. Less time is spent on the project plan in XP as change is expected, thus more effort is spent developing than planning, which produces initial software quicker. These results in the project planning process area add to the argument that the CMMI-Dev is overly bureaucratic and is heavily documented (Dickerson 2001).

To improve the mapping between the planning game practice and the project planning process area, the planning game practice would need to focus more on resources by conducting an initial review of determining what staff skills, technologies and

materials would be required to complete the project. This would then lead to discovering project budget, schedule and identifying plausible risks to the project. The project plan process area could then be used by the planning game practice to monitor project resources after each iteration, to check whether resources had deviated from the plan.

4.6 Software Quality Assurance

4.6.1 Process and Product Quality Assurance & Coding Standards

Table 19: Process and Product Quality Assurance & Coding Standards Mapping Results

Specific Goal Result No	Process Area (PA) & Specific Goal (SG)	Specific Practices (SP) Mapping Results	XP Practice
PPQA45	**PA:** Process & Product Quality Assurance **SG1:** Objectively Evaluate Processes and Work Products	2/2 Unsatisfied	Coding Standards
			Further Satisfy Goal TDD
PPQA46	**PA:** Process & Product Quality Assurance **SG2:** Provide Objective Insight	2/2 Unsatisfied	Coding Standards
			Further Satisfy Goal Pair Programming
Coverage Calculation	**Process & Product Quality Assurance**	**100% Unsatisfied**	**Coding Standards**

Unsatisfied Mapping Results

PPQA45 Table 19 shows that the coding standards practice does not satisfy any specific practice of the process and product quality assurance process area. Specific goal one, objectively evaluates processes and work products, focusing on the processes that the organisation have in place to create work products and objectively evaluate these processes against process descriptions and standards. The coding standards practice solely focuses on coding between developers to maintain a high standard or consistency throughout development and does not map to specific goal one.

PPQA46 Specific goal two is also fully unsatisfied by the coding standards practice. Evaluations are performed to discover noncompliance issues which reflect a lack of adherence to applicable standards, process descriptions and procedures in this specific practice. The coding standards practice does not satisfy this specific goal because of the sole focus on producing a high standard of coding and not considering any processes or techniques the developers use to code.

Further Analysis

To improve the mapping between the coding standards practice and the process and product quality assurance process area, it would be beneficial for the coding standards practice to establish a process to meet high coding standards so that future projects could learn from previous projects errors. Objective insight by the other pair programmer could be added to improve the mapping included in the pair programming practice. This would allow another programmer, other than the one developing the system, to give their opinions on the standard of the code produced. The test driven development practice could also be used to improve the mapping as quality is built into the code by testing first, which focuses the developer on the client requirements and by testing frequently the quality of the code is being maintained.

4.7 Evaluation Conclusion

This section summarises the overall results produced previously into two manageable tables. An overall mapping results table that shows ratings for the individual specific practices of a process area in explicit detail and a process area coverage results table showing overall percentages of how unsatisfied, partially or satisfied the process area is with its associated XP practice. This allows results analysis to be conducted that reveals patterns and relationships to be discussed in the final chapter of the study.

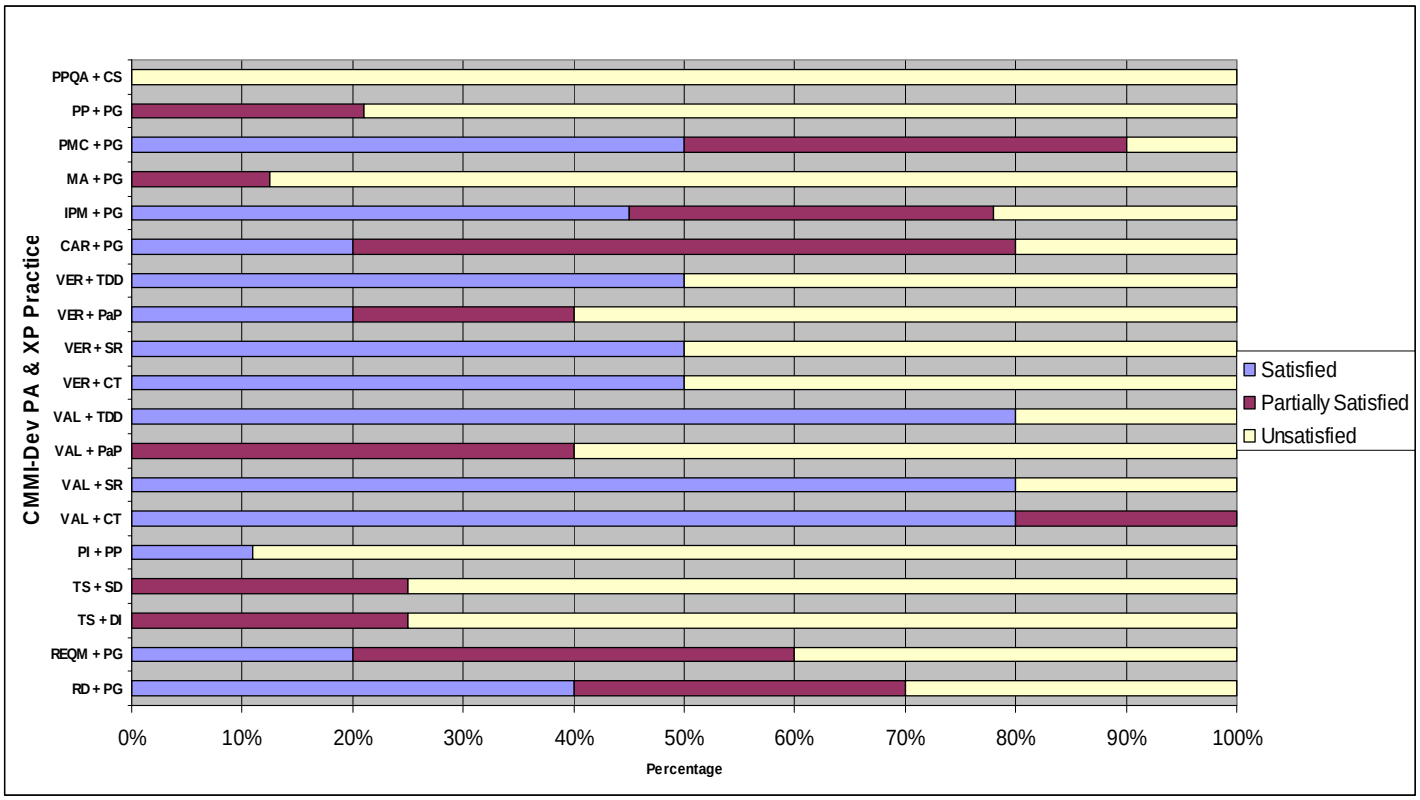
Table 20: Overall Mapping Results

Development Activity	Process Area	SG	SP 1	SP 2	SP 3	SP 4	SP 5	SP 6	SP 7	Overall Mapping
Soft R & A	RD	SG1	PS	PS	-	-	-	-	-	**PS**
-	RD	SG2	U	U	U	-	-	-	-	**U**
-	RD	SG3	S	PS	S	S	S	-	-	**PS**
-	REQM	SG1	PS	S	PS	U	U	-	-	**PS**
Design	TS	SG1	US	U	-	-	-	-	-	**U**
-	TS	SG2	PS	U	U	U	-	-	-	**PS**
-	TS	SG3	PS	U	-	-	-	-	-	**PS**
Implementation	PI	SG1	U	U	U	-	-	-	-	**U**
-	PI	SG2	U	U	-	-	-	-	-	**U**
-	PI	SG3	U	S	U	U	-	-	-	**PS**
Validation	VAL	SG1	S	S	PS	-	-	-	-	**PS**
-	VAL	SG2	S	S	-	-	-	-	-	**S**
Verification	VER	SG1	S	S	U	-	-	-	-	**PS**
-	VER	SG2	U	S	U	-	-	-	-	**PS**
-	VER	SG3	S	S	-	-	-	-	-	**S**
Soft Proj Man	CAR	SG1	S	PS	-	-	-	-	-	**PS**
-	CAR	SG2	PS	PS	U	-	-	-	-	**PS**
-	IPM	SG1	S	S	PS	U	PS	U	-	**PS**
-	IPM	SG2	S	PS	S	-	-	-	-	**S**
-	MA	SG1	PS	U	U	U	-	-	-	**PS**
-	MA	SG2	U	U	U	U	-	-	-	**U**
-	PMC	SG1	PS	S	PS	U	PS	S	S	**PS**
-	PMC	SG2	S	S	PS	-	-	-	-	**PS**
-	PP	SG1	PS	PS	U	PS	-	-	-	**PS**
-	PP	SG2	U	U	U	U	U	U	U	**U**
-	PP	SG3	U	U	U	-	-	-	-	**U**
QA	PPQA	SG1	U	U	-	-	-	-	-	**U**
-	PPQA	SG2	U	U	-	-	-	-	-	**U**

Table 20 shows the overall mapping results of the XP practices to the process area under the allocated generic development activity. The results do not take into consideration the XP practices that could further satisfy the process area due to being analysed under a generic software development activity but previous discussion has identified where possible inclusion could be mapped. Table 21 shows the process area coverage results between each CMMI-Dev process area and XP practice revealing the percentage of satisfied, partially satisfied and unsatisfied specific practices in the mapping.

Process Area Coverage Results

Table 21: Process Area Coverage Results



These tables show an in-depth analysis into the quality practices of XP and relations between the quality process areas of the CMMI-Dev. These results could be used by organisations to make sound judgements on improving their own software development processes and practices. Both tables summarise the results of this chapter in detail and allow the final discussions and conclusions section of the report to draw on the identified relationships between XP and the CMMI-Dev.

5.0 Final Discussion & Conclusions

This section of the report will present the final overall conclusions that can be drawn from the results of the evaluation method detailed in the previous chapter of this project. A brief summary of the project work will also be discussed within this chapter and will conclude with what future work could be undertaken following the results of this project.

5.1 Brief Summary Of Project

The purpose of the categorisation study was to identify how successfully the quality practices of XP could be incorporated into the CMMI-Dev for Development Version 1.2. By applying the quality practices of Extreme Programming to the CMMI, the study provides further research into improving the manner in which software is developed with quality control.

A review was conducted into the generic activities of software development that revealed common development activities that are standard in the majority of software development projects. An investigation was then conducted into Extreme Programming's quality practices and their relation to the general software development activities. The characteristics and quality process areas of the CMMI-Dev were then analysed in order to provide an understanding into how they related to a specific generic development activity. By analysing the CMMI-Dev process areas and the XP practices in relation to common software development activities, a framework was developed that showed each process area of the CMMI-Dev and XP practice grouped under common software development activities, as shown in Figure 6, pg 44. This allowed practices and process areas that did not relate to the common framework to be eliminated from the study before proceeding to the experiment. The final experiment analysed 12 process areas of the CMMI-Dev and 8 practices of XP, which were discovered as relevant to this study.

5.2 Final Discussion Of Results

In this section, the results of the study will be discussed in relation to the identified hypothesis that only certain practices of XP can be introduced to the disciplined, traditional Quality Management System of the CMMI-Dev. The results are discussed within the generic development activities of a project and include individual conclusions and reference to how they link to the work of others.

Software Requirements & Specification Activities

Table 22: Software Requirements & Specifications Mappings

Process Area	Specific Goal	Overall Mapping	Critical Analysis	Fully Satisfied %
Requirements Development	SG1	PS	Focuses profoundly on client requirements. Could improve by performing a pre-activity of analysing the product requirements. XP practice has the building blocks, just needs to be adapted to fully satisfy the goal.	**40%**
	SG2	U	Sets out to understand business and process requirements which cannot be applied due to XP not discovering requirements other than the clients.	
	SG3	PS	Highly maps with 4/5 SPs being fully satisfied.	
Requirements Management	SG1	PS	The XP activities could be adapted in a way that structures the management of requirements from analysing to tracing. This would increase the mapping.	**20%**

Table 22 shows results from the software requirements and specification mappings that the XP practice and the CMMI-Dev process areas have a low overall coverage rating. Both process areas of development requirements and management requirements produced low results of 40% and 20% respectively of being fully satisfied with the planning game practice. One of the reasons there was such a low mapping was due to the XP practice solely developing based on the client requirements and not considering developing requirements based on the business or product requirements.

To increase the mapping rating between the planning game practice and the requirements development process area, it would be useful for the XP practice to have a pre-activity of analysing the product requirements before developing the client requirements. This would allow an organisation's development team to understand current processes and organisational workings revealing other important requirements instead of solely focusing on the client. The mapping between the requirements management process area and the planning game practice could also be further mapped by including a requirements management process or structure. This would deal with monitoring and recording requirements and would be beneficial to an organisation to be able to trace requirements through the project to check if they are being satisfied. In a study by Pikkarainen and Mäntyniemi (2006), which involved mapping agile practices to the CMMI in a real life organisation, the requirements management specific goal had problems. They found that when mapping agile practices to the CMMI requirements management specific goal, their results showed that discovering requirements was acceptable but problems resided in managing the requirements. The results in this study support their findings in that XP

would map further if it were to adopt a more structured approach to managing the requirements.

The expected results of this section, prior to conducting the mappings was that there would be a higher coverage rating between the CMMI-Dev process areas and the XP practice involved. Both CMMI-Dev process areas and the XP practice aim to discover, justify and validate why requirements are needed but the study has revealed that there are differences in the way the CMMI-Dev and XP develop and manage requirements. This may be due to a clash in paradigms where the CMMI-Dev prefers to have all requirements upfront in a waterfall approach where as XP discovers more client requirements as each iteration progresses.

When reflecting on the hypothesis, the software requirements and specification results support the hypothesis that only certain XP practices will map to the CMMI-Dev. The XP planning game practice does not fully satisfy the requirements development or management process areas of the CMMI-Dev but XP could be modified to further satisfy the mapping.

Design Activities

Table 23: Technical Solution Mappings

Process Area	Specific Goal	Overall Mapping	Critical Analysis	Fully Satisfied %
Technical Solution	SG1	U	Alternative design solutions to be identified and analysed is required. Cannot be satisfied by XP as product is constantly evolving and being improved by each iteration. Would also compromise the speed at which XP produces an initial piece of working software.	**0%**
	SG2	PS	XP does not develop the full design upfront before coding starts. This is based upon a clash of paradigms where the CMMI-Dev follows a more traditional waterfall approach and XP an incremental methodology of development.	
	SG3	PS	Implement the design. Design techniques such as refactoring can be used during coding but this goal requires the design to be fully coded. XP works on the design throughout development instead of producing it all upfront. Another clash of methodologies.	

Table 23 shows results from the design activities that the XP practices and the CMMI-Dev process area have a low overall coverage rating. Both XP practices of simple design and design improvement fail to fully satisfy any process goal of the technical solution process area. The main reason behind such an unsuccessful mapping was due to the technical solution requiring a large amount of upfront design to be completed before the implementation activities commenced where as the XP practices of simple design and design improvement are continuous throughout the life of the project working on an ever-evolving and refined design.

To improve the mapping rating between the design improvement practice and the technical solution process area, it would be beneficial for an organisation to have software tools that automatically reverse engineer the product designs. This could produce the documentation that is required by the technical solution process area and further map the design improvement practice and technical solution process area. This would allow an organisation's development team more time to focus on design and implementation activities than having to produce supporting documentation to justify designs. In the study by Anderson (2005), they found that by introducing a combination of integrated tooling and an overall agile approach into a plan driven process design, it reduced the overhead in the process by around 85% supporting the view that an agile approach can be adapted to suit a plan driven process area. To improve the mapping between the technical solution process area and the simple design practice, techniques like refactoring could be transferred to the technical solution process area and used to focus more on design during implementation. This would make the design more flexible to changes as they occur in the implementation phase, which could possibly reduce re-working and maintenance costs for an organisation. This would allow an organisation to embrace change rather than reject, control or suffer from change which studies by Willoughby (2005) have shown the CMMI often enforces.

The expected results of the design activities, prior to conducting the mappings was that there would be a low coverage rating between the CMMI-Dev process area and the XP design practices. This was due to different design approaches, the complete upfront design of the traditional CMMI-Dev compared to the continuously ever-evolving XP design approach. The results confirmed this prior expectation but there are improvements that can be made to the mappings such as the automated tools that would produce more documentation for XP and the refactoring technique being incorporated further into the CMMI-Dev process area to help produce a flexible design more open to change.

When reflecting on the hypothesis, the design results support the hypothesis that only certain XP practices will map to the CMMI-Dev as the two design practices of XP fail to map. The XP planning game practices of simple design and design improvement do not satisfy the technical solution process area of the CMMI-Dev but there are modifications such as tools and reverse engineering that could be adapted to the XP practice to improve the mapping results.

Implementation Activities

Table 24: Implementation Mappings

Process Area	Specific Goal	Overall Mapping	Critical Analysis	Fully Satisfied %
Product Integration	SG1	U	Product Integration establishes and maintains the environment for the system. XP assumes the environment is established as the main focus is solely on developing the software.	**11%**
	SG2	U	Specifies what needs to be completed to integrate the system where as XP identifies how the implementation should be performed.	
	SG3	PS	Focuses more on evaluation and testing within the client's environment. Could be more fuller satisfied by adding XP Validation &Verification activities.	

Table 24 shows the results from the implementation activities that the pair programming XP practice has a very low coverage rating of 11% with the product integration process area. One of the reasons behind such an unsuccessful mapping was due to the pair programming practice focusing solely on problem solving and physical coding where as the product integration process area focuses on managing and supporting the process of implementation.

If an organisation using XP were investigating the CMMI-Dev to integrated quality practices, it would be beneficial to have an area that was a hybrid of both. This would be more successful with the product integration process area specifying what needs to be completed to integrate the system and the pair programming practice providing how the implementation should be performed. To improve on the mapping, validation and verification activities such as client tests, TDD and small releases would all contribute to a higher mapping in specific goal three as it focuses on evaluating and testing components.

The expected results of the implementation activities, prior to conducting the mappings was that there would have been a greater success in mapping the product integration process area and pair programming practice as they were the only activities identified during the literature review that related to implementation. As they were the only implementation activities, it was expected that there would have been a higher commonality between both than the results produced. When reflecting on the hypothesis, the implementation results support the hypothesis that only certain XP practices will map to the CMMI-Dev as the pair programming practice of XP fails to map.

Validation

Table 25: Validation Mappings

Process Area	Specific Goal	Overall Mapping	Critical Analysis	Fully Satisfied %
Validation	SG1	PS	High mapping with XP practices as all SPs are fully satisfied except one being partially satisfied. Client tests, TDD, small releases allow requirements of the system to be checked that they conform to the requirements specification.	**80%**

Table 25 shows results from the validation mappings that the XP practices and the CMMI-Dev process area have a high overall coverage rating. The XP practices of client tests, small releases and test driven development all have an 80% satisfied coverage rating with the validation process area. The main reason behind such a successful mapping is due to testing and releasing software in small phases throughout development. By involving the client in the tests, their expectations are more fully understood by the development team and by testing frequently, the requirements are being monitored continuously that they conform to the specification.

The pair programming practice when mapped to the validation process area produced a fully unsatisfied mapping with no process goal able to be fulfilled. This was expected however, due to the pair programming practice mainly focusing on implementation, bringing the product from design to a working piece of software where as the validation process area has the objective of specifying whether the product will fulfil its intended use.

The study by Fritzsche and Keil (2007), supports these results that the validation process area maps highly due to XP's intensive testing practices of client tests, test driven development and small releases by emphasising the important relation between validation and testing. The mapping between the validation process area and small release practice was expected to have a high rating due to building in small increments where the software produced frequently during the project could be tested to check if it conforms to the client's requirements. The analysis from the testing results could then be taken and incorporated into the next iteration to improve or change functionality depending on what the client requests. The successful mapping between the client tests practice and the validation process area was also to be expected. By involving the client in the tests, their expectations are more fully understood by the development team thus there is more chance the requirements will be correct and validated. The client tests satisfy the validation specific goals, as the requirements of the system are checked that they conform to the requirements specification as emphasised by the validation process area. The validation process area and test driven development was expected to produce a high mapping rating due to the continuous testing that the test driven development practice

enforces throughout the project. This helps to satisfy the validation aim of proving that a piece of code or functionality fulfills its intended use.

These results are particularly useful to an organisation that requires improvement on their validation activities of software development. With such a high mapping between the agile client tests, small releases and test driven development practices of XP with the reliant and traditional CMMI-Dev validation process area, an organisation could identify which XP practice would improve their own quality management framework.

When reflecting on the hypothesis, the validation results support the hypothesis that certain XP practices will map to the CMMI-Dev as the three out of four validation practices of XP highly map. The XP practices of client tests, small releases and test driven development highly satisfy the validation process area of the CMMI-Dev and could fully map the CMMI-Dev process area if slight modifications were made.

Verification

Table 26: Verification Mappings

Process Area	Specific Goal	Overall Mapping	Critical Analysis	Fully Satisfied %
Verification	SG1	PS	Requires organisational procedures and criteria to be established in order to verify products which XP does not do.	**57%**
	SG2	PS	Peer reviews are required to be prepared, conducted and reviewed that are not satisfied except to an extent by the pair programming practice of XP.	
	SG3	S	Verification results are obtained by the XP activities where they can be analysed to check whether they achieve the project requirements and objectives.	

Table 26 shows the results from the verification activities that the XP practices and the CMMI-Dev process area have a mediocre coverage rating. The XP practices of client tests, small releases and test driven development all have a 50% satisfied coverage rating with the verification process area and the pair programming adds to the fully satisfied percentage to make it up to 57%. The main reason behind such results is due to the focus on testing, which justifies to the developers and client that the product is operating as expected and by releasing in small phases, the piece of software can be frequently analysed to verify that it meets the specification. Any requirements that are unclear

during development can be clarified by the client involved in the testing, to allow the developer to gain a thorough understanding of the client's needs. This allows the product or piece of functionality to demonstrate that it fulfills the project requirements and objectives.

To further increase the mapping between the verification process area and XP verification practices, the pair programming practice could be integrated. Vriens (2003) findings show that pair programming can act as peer reviews supporting this study's results that XP can be adapted to more fully satisfy the verification process area of the CMMI-Dev. An organisation using the CMMI-Dev, may use these results to conclude that the validation practices of XP alone would not improve their current process area of verification but could use aspects of the pair programming practice to become more agile in development.

These results are particularly useful to an organisation that currently use XP and want to improve on their verification activities of software development. The results show that the agile client tests, small releases and test driven development practices can be mapped to an extent to the traditional CMMI-Dev validation process area, but could further be improved by incorporating the pair programming practice.

When reflecting on the hypothesis, the verification results support the hypothesis that certain XP practices will map to the CMMI-Dev. The XP practices of client tests, small releases and test driven development, that include validation activities, could further map to the validation process area of the CMMI-Dev with the support of the pair programming practice.

Software Project Management

Table 27: Software Project Management Mappings

Process Area	Specific Goal	Overall Mapping	Critical Analysis	Fully Satisfied %
Casual Analysis & Resolution	SG1	PS	XP needs to produce supporting documentation when addressing the causes of the defects and evaluating the effect of the changes to track why the defect occurred, monitoring the effect of resolving the defect and trace the defect through development. Tools that could automatically generate defect data would also further improve the mapping between XP and the casual analysis and resolution process area.	**20%**
	SG2	PS		

Integrated Project Management	SG1	PS	The documentation required to be produced to support the specific practices fails in the mapping and is one of the main reasons why 33% of the specific practices remain partially satisfied. XP emphasises working software over documentation where as the integrated project management process area produces a large amount of supporting documentation that is not produced during XP management activities.	45%
	SG2	S		
Measurement & Analysis	SG1	PS	XP is applied in a project to project basis, one project at a time, where as this CMMI-Dev framework would be used for every project in the same rigorous manner regardless of project variation. To increase the mapping of the measurement and analysis process area with XP, there would need to be more project management involved in the XP practices.	**0%**
	SG2	U		
Project Management & Control	SG1	PS	40% of the project monitoring and control process area is partially satisfied by XP. To improve the mapping between the CMMI-Dev process area and the XP practice, the planning game practice requires more project management to allow project resources to be monitored as the project progresses. The reason the 40% are partially and not fully satisfied is due to the planning game practice not monitoring or documenting them as the development progresses. This would allow deviations from the plan to be dealt with at regular reviews and action to be taken in the next planning game iteration.	**50%**
	SG2	PS		
Project Planning	SG1	PS	Clash of cultures between XP and the CMMI-Dev as XP embraces change, which results in little planning at the beginning of the project. Less time is spent on the project plan in XP as change is expected, thus more effort is spent developing than planning, which produces initial software quicker.	**0%**

Table 27 shows the results from the project management activities that the XP planning game practice and the five CMMI-Dev process areas produce very low coverage ratings. The planning game practice fails to fully satisfy any specific goal of the measurement and analysis process area and the project planning process area. These results are support by Fritzsche and Keil (2007) in their study that confirms that XP provides no specific guidelines for the tasks of measurement and analysis. The results also support the

findings that XP lacks on long-range planning with the focus instead on short-range planning for each iteration as discussed in the report by Kane and Ornburn (2002). The other process areas of casual analysis and resolution, integrated project management, and project management and control mapped similarly low with 20%, 45% and 50% respectively.

The results suggest that XP is applied in a project to project basis where as the CMMI-Dev framework can be used for every project in the same rigorous manner regardless of project variation. The low mapping by the project management activities of XP shows that XP does not plan for future projects by trying to learn from previous mistakes. There is no structure to store or record information from previous projects that may be useful in the future and justifies Jefferies (2000) article discussion that if XP was being conducted throughout an entire large organisation, more measurement would be needed than in a single one off project. To map more successfully to the project management activities of the CMMI-Dev, the planning game practice needs to produce supporting documentation when addressing the causes of defects and evaluating the effect of changes. Documents such as action proposals would allow an organisation to track why the defect occurred, monitoring the effect of resolving the defect and could be used to trace the defect through development to apply lessons learnt in future projects. By not tracking development measurements, this could result in the code lacking quality which supports the results in the study by Fitzgerald, Hartnett & Conboy (2006) in which after combining Agile methods into their quality framework, quality within the code had reduced. By introducing tools that would automatically record, document and trace defect data, it would improve the mapping between the planning game practice and the project management activities as it would ease the burden of producing large amounts of documentation.

The unsuccessful coverage rating between the planning game practice and the project management activities of the CMMI-Dev was to be expected as XP embraces change, which results in less planning within each project. With the focus more on developing than planning, the project delivers working software earlier and is seen by the client that progress is being made. The low mapping is also due to a clash of cultures between XP and the CMMI-Dev as XP embraces change, which results in little planning at the beginning of the project where much change is handled in the first iteration. Less time is spent on the project plan in XP as change is expected, thus more effort is spent developing than planning, producing working software quicker and resulting in a low mapping to the meticulous project planning of the CMMI-Dev.

These results are particularly useful to an organisation currently using XP and want to have a more structured software development framework. They can view these results, and identify where their XP practices needs to be modified to satisfy the CMMI-Dev. The results show that XP lacks in project management and fails to satisfy the project management process areas of the CMMI-Dev. Automatic defect and documentation tools would further satisfy the mapping relieving the pressure on producing a large amount of documentation and allowing more time to plan. When reflecting on the hypothesis, the project management results support the hypothesis that only certain XP practices will

map to the CMMI-Dev with the planning game practice of XP unsuccessfully mapping to the project management process areas of the CMMI-Dev.

Software Quality Assurance

Table 28: Software Quality Assurance Mappings

Process Area	Specific Goal	Overall Mapping	Critical Analysis	Fully Satisfied %
Product and Process Quality Assurance	SG1	U	Objectively evaluate organisational processes against process descriptions and standards. This is more of an organisational self improvement practice that has nothing to do with XP.	**0%**
	SG2	U	Evaluations are performed to discover noncompliance issues which reflect a lack of adherence to applicable standards, process descriptions and procedures. The coding standards practice does not satisfy this specific goal because of the sole focus on producing a high standard of coding and not considering any processes or techniques the developers use to code.	

Table 28 shows the results that the coding standards practice does not satisfy any specific practice of the process and product quality assurance process area. The process and product quality assurance process area objectively evaluates processes and work products, focusing on the processes that the organisation have in place to create work products where as the coding standards practice solely focuses on coding between developers. The results show that this does maintain a high standard and consistency of coding throughout development but cannot be related to other areas of XP as it solely focuses on coding. A study by Nawrocki, Bartosz, and Wojciechowski (2001) into creating an agile equivalent of the CMMI, found that there are almost no relations between XP practices concerning coding and CMMI process areas which supports the 0% mapping of SQA activities found by this study. This emphasises that XP lacks in software quality assurance activities by only having one practice able to contribute to this area of software development although the test driven development practice could improve the mapping. The test driven development practice could improve the mapping as quality is built into the code by testing first, which focuses the developer on the client requirements and by testing frequently the quality of the code is being maintained.

To improve the mapping between the coding standards practice and the process and product quality assurance process area, it would be beneficial for the coding standards practice to establish processes to provide evidence that higher coding standards are being met and that future projects could learn from previous project errors.

The unsuccessful coverage rating between the coding standards practice and the process and product quality assurance process area of the CMMI-Dev was to be expected, as XP focuses more on product quality than process quality. In a study by Wilkie, McFall and McCaffery (2005), they found that small organisations focus more on product quality assurance than on process quality assurance. The results of this study support their findings, as XP's quality assurance activities do not map to the CMMI-Dev as they focus on the quality of the product rather than on the quality of the processes to create the product. This supports the theory that many of the CMMI process areas and goals that are concerned with process quality are perceived as costly and time-consuming activities as organisations may focus more specifically on product quality which agile practices emphasise (Lycett et al. 2003).

5.2 Project Critique

In this section of the evaluation, the project is critically appraised by discussing how successfully the original project aims were achieved. Any problems that arose during the project are discussed and reflection is provided on how the project could have been improved if done differently.

It is important to critical appraise whether the project met its aims and objectives as outlined in section 1.2, page 11. The first aim was to carry out an investigation into the generic activities of a standard software development process, which was completed successfully through analysis of each development activity. The second and third objectives were to review XP's quality practices and review the characteristics and quality process areas of the CMMI-Dev and then relate both to the general software development activities. These objectives were completed through an extensive literature review that justified their relations with quality and the standard development lifecycle. The fourth objective was to identify the CMMI-Dev process areas and the XP practices that could be included under a particular generic activity for software development. This was performed to a high standard and resulted in a framework, as shown in Figure 6, pg 44, which laid the foundation for the experiment. The final aim to be completed was then to incorporate the XP quality practices into the CMMI-Dev process areas under the generic development framework.

Project Strengths

This study has provided an in depth look at the quality practices and process areas of the CMMI-Dev and XP, with a view of adding to the understanding of relationships between both to further improve the software process improvement field. The project is theoretically a well grounded piece that fulfils the first, second, third and fourth objectives by carrying out an in depth review of relevant literature and other information sources. Firstly it draws upon research about standard development activities that are common in the majority of projects with reference to why they are important. Secondly, the study covers the XP practices and CMMI-Dev using in depth literature research to further understand their quality approaches to software development and critically

analysing their relationships to the standard development activities. By performing critical analysis after the in depth literature review within a research context, the study was able to identify a framework of CMMI-Dev process areas and XP practices that could be included under a particular generic activity, satisfying the fourth aim. The literature review identified areas where relationships may be discovered between the generic development activities of both the CMMI-Dev and XP satisfying the final objective that referred to the experiment.

Mappings between process areas and practices were carried out under an analytical approach using the official CMMI for development Version 1.2 documentation. In using this methodology, the study was able to extract data and explore relationships between XP and the CMMI-Dev under the framework established by the in depth literature review. The mappings of the experiment were then subjected to data analysis in order to establish relationships by letting theory emerge directly from the mapping results. The high level data analysis tables shown in the results chapter, are able to provide insight into relationships between the agile XP practices and the traditional CMMI-Dev process areas fulfilling objective five. By allowing coverage ratings to be calculated, the study is able to successfully give an overall analysis of how successful XP practices can map to the relevant CMMI-Dev process areas under a generic development activity.

Project Weaknesses

A theory based approach to analysing the relationships between the CMMI-Dev process areas and XP practices produces a large amount of qualitative data to analyse. This can be of benefit in further understanding the relationships between agile development and quality frameworks and could be possibly used in future projects. Although a theory approach has benefits, it would have been beneficial to have the experience of practitioners who are familiar with the CMMI-Dev framework and agile practices to develop software. This would have given first hand experience and inside knowledge to the study, instead of being solely literature based. The difficulties encountered in recruiting practitioners for the study would be that they may be reluctant to share their views on failures or successes of the CMMI-Dev or XP even though this project was not concerned with exploring negative outcomes of the CMMI-Dev and XP.

What could not be included in the study was the analysing of the configurations management and risk management generic activities. The initial trial mapping revealed that there would have not been enough time to complete the study if these two generic activities were mapped between the CMMI-Dev and XP practices thus they were excluded from the study. If the study was to be performed again, more time would have been allocated to the experiment as by increasing the result size, the study is making more of a contribution to providing vital information to the field of process improvement between agile and quality frameworks.

5.3 Further Work

This study has revealed issues in project management for the XP practices as only one practice could be considered in the generic project management activities, which was the planning game practice. The traditional management framework in an organisation has a structure that includes management of organisational processes, which is a significant omission in the agile approach thus further work into agile project management would be beneficial in the first step to solving the bridge between organisational processes and XP practices.

Further work that could build on this study, would be to research into the possibility of having an agile version of the process areas of the CMMI-Dev. This could be useful to organisations that have agile development practices in place and are considering implementing the CMMI-Dev to achieve a higher level of company status that the CMMI-Dev maturity assessments can provide.

Another project could be completing the remaining CMMI-Dev process areas and XP practices that can be included under the risk management and configuration management generic activities. This would provide a complete study investigating relationships between the appropriate XP practices and CMMI-Dev process areas under their identified generic activities of software development.

It would be of interest to the software process improvement industry for further work to be conducted into how successful the mappings of this theory based study are when compared to the results in a real life scenario. An organisation that currently uses XP or the CMMI-Dev would need to be identified and the same generic framework for this study could be applied. This would reveal further relationships between the CMMI-Dev process areas or XP practices and would give further insight into software process improvement in a real life context.

5.4 Conclusions

The project has successfully identified relationships between XP practices and process areas of the CMMI-Dev under generic software development activities, highly satisfying and focusing on the research question. The study was conducted in a proficient and articulate manner through a detailed literature research and methodology experiment that captured insightful and new information into the software process improvement field.

The findings of this study would be particularly useful to organisations that currently apply XP practices and are interested in the CMMI-Dev. The results of this study would allow the company to analyse their current practices and develop an initial understanding to what XP practices relate to the CMMI-Dev process areas and how successful they map. This study could also interest organisations currently using the CMMI-Dev to further their knowledge on blending Agile development with their quality management framework.

This study has revealed that when XP is mapped to the CMMI-Dev under generic activities of software development, areas such as design, implementation and SQA unsuccessfully map. The low mapping between project management activities in XP and the CMMI-Dev, highlights the lack of project management in XP and supports the objection (Paulk 2001 ; Lycett et al. 2003) to using XP in large organisations because it barely touches the management and organisational issues that the CMMI emphasises. However, the continuing argument that suggests that XP cannot be used in the rigorous, process intensive CMMI is not supported by this study. XP uses well defined techniques and practices and has shown in this study that it highly satisfies the CMMI-Dev in the verification and validation activities by emphasising testing and client involvement. The CMMI-Dev informs organisations what to do where as XP is a set of best practices that has information on how to develop software thus by using a combination of both, developing software in the future may be more successful. This methodical and thorough study can thus conclude that the traditional quality framework of CMMI-Dev and agile best practices of XP can complement one another in certain areas of software development.

References

Abrahamsson, P., Warsta, J., Siponen, M. and Ronkainen, J. (2003) New Directions on Agile Methods: a comparative analysis, *25th International Conference on Software Engineering*, 2003, IEEE Computer Society, p.244.

Anderson, D.J. (2005) Stretching agile to fit CMMI level 3 - the story of creating MSF for CMMI/spl reg/ process improvement at Microsoft corporation, *Agile Conference, 2005. Proceedings*, 2005, p.193-201.

Baker, S.W. (2006) Formalizing agility, part 2: how an agile organization embraced the CMMI, *Agile Conference, 2006*, 2006, p.8.

Baskerville, R., Ramesh, B., Levine, L. & Pries-Heje, J. 2006. High-Speed Software Development Practices: What Works, What Doesn't. IT Professional Magazine, 8(4), 29.

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

Boehm, B. and Turner, R., 2004. Balancing Agility and Discipline: A Guide for the Perplexed, 1st ed. UK: Addison Wesley.

Boehm, B.W. 1984. Verifying and Validating Software Requirements and Design Specifications. IEEE Software, 1(1), 75-88.

Bollinger, T.B. 1991. A critical look at software capability evaluations. Software, IEEE, 8(4), 25-41.

Buede, D.M. 1997. Developing originating requirements: defining the design decisions. IEEE Transactions on Aerospace and Electronic Systems, 33(2), 596-609.

Buede, D.M. 1997. Integrating requirements development and decision analysis. 1997 IEEE International Conference on Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'., 2, 1574-1579 vol.2.

Cane, A., 2007. Flexibility takes over from plans in an agile world. , p. 6, .

Canfora, G., Cimitile, A., Garcia, F., Piattini, M. & Visaggio, C.A. 2007. Evaluating performances of pair designing in industry. The Journal of Systems and Software, 80(8), 1317.

Carnegie Mellon University. 2008. What is CMMI?[online]. Available from: http://www.sei.cmu.edu/cmmi/general/index.html#adoption [01/21 2008].

Chong, X. (2006) The Process Model of Organization Innovation, ZHAO SHENG-BIN, ed. In: *International Conference on Management Science and Engineering, 2006. ICMSE '06. 2006*, 2006, ICMSE, p.1302-1306.

Chrissis, M.B., Konrad, M. and Shrum, S., 2006. CMMI(R): Guidelines for Process Integration and Product Improvement, 2nd ed. Canada: Addison-Wesley.

CMMI PRODUCT TEAM, 2006. *CMMI for Development Version 1.2*. Pittsburgh, PA: Software Engineering Institute.

CMMI PRODUCT TEAM, 2002. *Capability Maturity Model Integration v1.1*. Pittsburgh, PA: Software Engineering Institute.

Crosman, P. 2007. Fast Lane -- Wall Street firms' insatiable appetite for new applications, delivered faster and with fewer bugs, is driving top technology executives at Merrill Lynch, HSBC and Mellon to pursue best practices for IT. Wall Street & Technology, 25(6), 32.

Damian, D. (2002) Global Software Development, *International Conference on Software Engineering (ICSE)*, 2002, ACM.

Damian, D. (2002) An industrial experience in process improvement: an early assessment at the Australian Center for Unisys Software, D. ZOWGHI, ed. In: *2002 International Symposium on Empirical Software Engineering. Proceedings.* 2002, p.111-123.

Dickerson, C. 2001. Build better software. InfoWorld, 23(33), Pg 12.

Dybå, T., Arisholm, E., Sjøberg, D.I.K., Hannay, J.E. & Shull, F. 2007. Are Two Heads Better than One? On the Effectiveness of Pair Programming. IEEE Software, 24(6), 12.

F. G. Wilkie, D. McFall & F. McCaffery 2005. An evaluation of CMMI process areas for small- to medium-sized software development organisations. Software Process: Improvement and Practice, 10(2), 189--201.

Fernández, W.D. 2005. Chapter 5. The grounded theory method and case study data in IS research: issues and design. In: D. Hart & S. Gregor, ed, Information Systems Foundations: Constructing and Criticising. 1st ed.

Fowler, M., Beck, K., Brant, J., Opdyke, W. and Roberts, D., 1999. Refactoring: Improving the Design of Existing Code , 1st ed. UK: Addison-Wesley Professional.

Fritzsche, M. and Keil, P. 2007. Agile Methods and CMMI: Compatibility or Conflict? E-Informatica Software Engineering Journal, 1(1).

Gibson, D.L., Goldenson, D.R., Kost, K. (SOFTWARE ENGINEERING INSTITUTE), 2006. *Performance Results of CMMI - Based Process Improvement.* Pittsburgh, PA: Carnegie Mellon Software Engineering Institute.

Goldenson, D.R., Jarzombek, J. & Rout, T. 2003. Measurement and Analysis in Capability Maturity Model Integration Models and Software Process Improvement. The Journal of Defense Software Engineering, (July 2003), 20-21, 22, 23, 24.

Grenning, J. 2001. Launching extreme programming at a process-intensive company. Software, IEEE, 18(6), 27-33.

Hall, E.M., 1998. Managing Risk: Methods for Software Systems Development, 1st ed. England: Addison Wesley.

Hinde, S. 2005. Why do so many major IT projects fail? Computer Fraud & Security, 2005(1), 15-17.

Hinkle, M.M. 2007. Software Quality, Metrics, Process Improvement, and CMMI: An Interview with Dick Fairley. IT Professional Magazine, 9(3), 47.

Hodge, R. (2002) Integrating reliability improvement modeling into practice-challenges and pitfalls, J. QUIGLEY, ed. In: *Annual Reliability and Maintainability Symposium, 2002. Proceedings.* 2002, p.158-164.

Huang, S. and Han, W. 2006. Selection priority of process areas based on CMMI continuous representation. Information & Management, 43(3), 297.

Huo, M. (2004) Software quality and agile methods, J. VERNER, ed. In: *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.* 2004, p.520-525 vol.1.

Institute of Configuration Management. 2008. Institute of configuration management [online]. Available from: http://www.icmhq.com/index.html [01/28 2008].

J. McGarry, D. Card, C. Jones, B. Layman, E. Bailey, J. Dean, et al, 2001. Practical Software Measurement: Objective Information for Decision Makers, Boston, MA: Addison Wesley.

Jeffries, R., 2001. What is extreme programming? XP Magazine, (http://xprogramming.com/xpmag/whatisxp.htm),

Jeffries, R., 2000. Extreme Programming and The Capability Maturity Model. XP Magazine, (http://www.xprogramming.com/xpmag/xp_and_cmm.htm)

Jiang, J.J., Klein, G., Hwang, H., Huang, J. & Hung, S. 2004. An exploration of the relationship between software development process maturity and project performance. Information & Management, 41(3), 279.

Juric, R. (2000) Extreme programming and its development practices, *Proceedings of the 22nd International Conference on Information Technology Interfaces, 2000. ITI 2000.* 2000, p.97-104.

Kahkonen, T. (2003) Digging into the fundamentals of extreme programming building the theoretical base for agile methods, ed. In: *Euromicro Conference, 2003. Proceedings. 29th*, 2003, p.273-280.

Kahkonen, T, Abrhamsson, P. (2004) Achieving CMMI Level 2 with Enhanced Extreme Programming Approach, *In proceedings of the 5th International Conference on Product Focused Software Process Improvement,* 2004, p.378 - 392.

Kane, D. and Ornburn, S. 2002. Agile Development: Weed or Wildflower? The Journal of Defense Software Engineering, 02/04/2008.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler et al,. 2001. Manifesto for agile software development [online]. Ward Cunningham. Available from: http://agilemanifesto.org/ [19/12 2007].

Konrad, M. & Over, J.W., 2005. Agile CMMI: No Oxymoron. The World of Software Development, (http://www.ddj.com/architect/184415287).

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., et al, 2004. Agile software development in large organizations. Computer, 37(12), 26-34.

Lycett, M., Macredie, R., Patel, C. & Paulk, R.J. 2003. Migrating agile methods to standardized development practice. Computer, 36(6), 79-85.

Marcal, Ana Sofia C 2007. Mapping CMMI Project Management Process Areas to SCRUM Practices. 31st IEEE Software Engineering Workshop, 2007. SEW 2007., , 13-22.

Martin, A. (2004) The XP customer role in practice: three studies, R. BIDDLE, ed. In: *Agile Development Conference, 2004,* 2004, p.42-54.

McCauley, R. 2001. Agile Development Methods Poised to Upset Status Quo. ACM SIGCSE Bulletin, 33(4), 14-15.

McConnell, S., 1996. Rapid Development, 1st ed. USA: Microsoft Press.

Nawrocki, J. (2001) Toward maturity model for extreme programming, B. WALTER, ed. In: *27th Euromicro Conference, 2001. Proceedings.* 2001, p.233-239.

Norausky, G.F. (2002) Casual Analysis and Resolution: A Business Driver at All Levels, NATIONAL DEFENSE INDUSTRIAL ASSOCIATION, ed. In: *National Defense Industrial Association 2nd Annual CMMI Technology Conference and User Group*, 2002.

Oates, B.J., 2006. Researching Information Systems and Computing, 1st ed. London: Sage Publications.

Ould, M.A., 1999. Managing Software Quality and Business Risk, 1st ed. Chichester, England: John Wiley & Sons.

Paulk, M.C., Chrissis, M.B., Weber, C.V. and Curtis, B., 1995. The Capability Maturity Model: Guidelines for Improving the Software Process, 2nd ed. Addison-Wesley Professional.

Paulk, M.C. 2001. Extreme programming from a CMM perspective. Software, IEEE, 18(6), 19-26.

Pikkarainen, M. and Mantyniemi, A. (2006) An Approach for Using CMMI in Agile Software Development Assessments: Experiences from Three Case Studies, *The 6th International SPICE Conference: Software Process Improvement and Determination*, 3-5th May, 2006.

Pressman, R.S., 2000. Software Engineering: A Practioner's Approach, 5th ed. England: McGraw-Hill.

Royce, W., 2002. CMM vs CMMI: From Conventional to Modern Software Management.

Rubinstein, R. 01/03/2007. Standish group report: There's less development chaos today [online]. Software Development Times. Available from: http://www.sdtimes.com/article/story-20070301-01.html [02/11/2007 2007].

Sangwan, R.S. 2006. Test-Driven Development in Large Projects. IT Professional, 8(5), 25-29.

Software Engineering Institute, 2007. *Capability Maturity Model Integration (CMMI) Version 1.2 Overview.* Pittsburgh, PA: Software Engineering Institute;.

Sommerville, I., 2006. Software engineering 8, 8th ed. UK: Addison-Wesley.

Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P. & Murphy, R. 2007. An exploratory study of why organizations do not adopt CMMI. Journal of Systems and Software, 80(6), 883-895.

Stojanovic, Z., Dahanayake, A. & Sol, H. 2004. Agile Development Methods and Component-Orientation: A Review and Analysis. In: K. Siau, ed, Advanced Topics In Database Research. 3rd ed. Hershey, PA: IDEA Group Publishers, 10-11, 12.

Strauss, A.L. and Corbin J, 1990. Basics of Qualitative Research: Grounded Theory Procedures and Techniques, 1st ed. Thousand Oaks: Sage.
Tomek, I. 2007. What I learned teaching XP [online]. Why Smalltalk. Available from: http://www.whysmalltalk.com/articles/tomek/teachingxp.htm [06/25 2007].

Tony Collins. 2007. Only A third of government IT projects succeed [online]. ComputerWeekly. Available from: http://www.computerweekly.com/Articles/2007/05/21/223915/only-a-third-of-government-it-projects-succeed-says.htm [15/10/2007 2007].

Turner, R. and Jain, A. (2002) Agile Meets CMMI: Culture Clash or Common Cause? D. Wells and E.L.A. Williams, eds. In: *Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - Xp/Agile Universe 2002 (August 04 - 07, 2002) Lecture Notes In Computer Science, vol. 2418*. 2002, Springer-Verlag, p.153-165.

Vantakavikra, P. (2007) Constructing a Process Model for Decision Analysis and Resolution on COTS Selection Issue of Capability Maturity Model Integration, ed. In: *6th IEEE/ACIS International Conference on Computer and Information Science, 2007. ICIS 2007*. 2007, p.182-187.

Vriens, C. (2003) Certifying for CMM Level 2 and IS09001 with XP@Scrum, *Proceedings of the Agile Development Conference, 2003. ADC 2003*. 2003, p.120-124.

Wang, M. (2007) An Intelligent Fuzzy Agent based on PPQA Ontology for Supporting CMMI Assessment, C. LEE, ed. In: *IEEE International Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007*. 2007, p.1-6.

Willoughby, M. 2005. Coder Be Agile, Coder Be Quick. Computerworld, 39(30), Pg36.

Wu, C., Wang, S. & Fang, K. 2007. The Effect of Organization Process Focus and Organizational Learning on Project Performance: An Examination of Taiwan's Companies. Portland International Center for Management of Engineering and Technology, , Pg. 1083 - 1089.

Zhenchang Xing. (2006) Refactoring Practice: How it is and How it Should be Supported - An Eclipse Case Study, ed. In: *22nd IEEE International Conference on Software Maintenance, 2006. ICSM '06*. 2006, p.458-468.

Glossary

CAR – Causal Analysis and Resolution
CM – Configuration Management
CMM – Capability Maturity Model
CMMI – Capability Maturity Model Integration
CMMI-Dev – Capability Maturity Model Integration for Product Development
CT – Client Tests
CS – Coding Standards
DI – Design Improvement
IPM – Integrated Project Management
MA – Measurement and Analysis
QPM – Quantitative Project Management
OPP – Organisational Process Performance
OPF – Organisation Process Focus
PG – Planning Game
PI – Product Integration
PMC – Project Monitoring and Control
PP – Pair Programming
PPQA – Process and Product Quality Assurance
REQM – Requirements Management
SAM – Supplier Agreement Management
SCM – Software Configuration Management
SD – Simple Design
SEI – Software Engineering Institute
SG – Specific Goal
SP – Specific Practice
SPI – Software Process Improvement
SR – Small Releases
SRS – Software Requirement Specification
SQA – Software Quality Assurance
TDD – Test Driven Development
TS – Technical Solution
V&V – Validation & Verification

Appendix A - Mapping Results

Table of Contents

List Of Tables

Software Requirements & Specification

Requirements Development & The Planning Game

XP Practice – Planning Game
CMMI-Dev Process Area – Requirements Development

Table 29: Requirements Development & The Planning Game Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Planning Game Practice
SG 1 Develop Customer Requirements SP 1.1 Elicit Needs SP 1.2 Develop the Customer Requirements **SG 2 Develop Product Requirements** SP 2.1 Establish Product and Product Component Requirements SP 2.2 Allocate Product Component Requirements SP 2.3 Identify Interface Requirements **SG 3 Analyse and Validate Requirements** SP 3.1 Establish Operational Concepts and Scenarios SP 3.2 Establish a Definition of Required Functionality SP 3.3 Analyse Requirements SP 3.4 Analyse Requirements to Achieve Balance SP 3.5 Validate Requirements	**(SG1)** Partially Satisfied **(SG2)** Unsatisfied **(SG3)** Partially Satisfied	Client presents desired features through use cases / storyboards. Developers produce cost estimates + requirement prioritisation. Initial schedule / plan produced before commencing iteration. Running software produced every 2-4 weeks. Customer involved in evaluating software. Feedback taken and incorporated into next 2-4 week iteration.

Specific Goal 1: Develop Customer Requirements

Goal Result: Partially Satisfied

Specific Practice 1.1: Elicit Needs

Practice Result: Partially Satisfied

- This specific practice collects all project requirements by proactively identifying all additional requirements not explicitly provided by the client, unlike the planning game practice of XP as it focuses solely on the client's requirements.
- Planning game practice does not reveal all requirements of the project but only those that are important to the client such as functionality and aesthetic requirements thus the XP planning game practice only partially satisfies this CMMI-Dev specific practice.
- A sub practice of this specific practice is to engage relevant stakeholders using methods for eliciting needs, expectations, constraints, and external interfaces. The XP practice does fully fulfill this requirement as methods such as use cases and storyboards are used by the development team and client to discover and extract requirements.

Mapping Key Points:

- XP practice does not reveal all requirements where as this specific practice, to be fulfilled, needs all additional requirements stated.
- The specific practice requires engagement by the stakeholders, which the XP practice fully satisfies as the client is continually involved.

Specific Practice 1.2: Develop The Customer Requirements

Practice Result: Partially Satisfied

- Documentation is not a priority of the planning game practice as working software is regarded more important thus does not fully satisfy the specific practice although some documentation may be produced.
- The development team in the planning game practice do produce cost estimates and prioritise requirements, which partially satisfies this specific practice.
- Conflicts between requirements are to be identified and resolved by this specific practice, which the planning game practice fully satisfies as the development team prioritise client requirements, identifying the most important to the project and disregarding any that are non-relevant or unrealistic.

Mapping Key Points:

- The XP practice does not produce all documentation where as the specific practice needs all specified and documented.
- The XP practice prioritises requirements eliminating conflicts and missing information that may have been accrued during the gathering of the requirements.

Specific Goal 2: Develop Product Requirements

Goal Result: Unsatisfied

Specific Practice 2.1: Establish Product and Product Component Requirements
Practice Result: Unsatisfied

- The product requirements are the expression of the client requirements in technical terms that can be used for design decisions. Other than the client requirements, this practice sets out to understand business and process requirements.
- The planning game practice does not satisfy this specific practice as the agile design is ever evolving as changes are continuously made thus there is no point establishing technical terms for them to change as they become obsolete after each iteration.
- XP focuses profoundly on the customer, developing around their requirements and allocates minimal resources on analysing the business and process requirements involved in the project.

Specific Practice 2.2: Allocate Product Component Requirements
Practice Result: Unsatisfied

- The requirements for product components of the defined solution include allocation of product performance and design constraints. Constraints and product performance are to be documented to every product and product component requirement, which the planning game cannot be applied to as it does not establish product or product component requirements.

Specific Practice 2.3: Identify Interface Requirements
Practice Result: Unsatisfied

- Interface requirements between components of the system are to be identified by this specific practice. The planning game practice does not satisfy this specific practice as interface requirements are not identified initially but will be developed as the project progresses.

Mapping Key Points:

- The planning game practice focuses mainly on the client's requirements and other requirements such as performance, product and interface are dealt with as development progresses. (Unsatisfied)

Specific Goal 3: Analyse and Validate Requirements

Goal Result: Partially Satisfied

Specific Practice 3.1: Establish Operational Concepts and Scenarios
Practice Result: Satisfied

- This specific practice is fulfilled by the planning game practice as use cases, storyboards and system scenarios are used to show how the system will interact with components, other systems and users.

Specific Practice 3.2: Establish a Definition of Required Functionality
Practice Result: Partially Satisfied

- The planning game practice satisfies this practice to a partial extent as understanding the client's interaction with the proposed or current system is the main focus of the development team early in development.
- Use cases, activity diagrams, storyboards and structured system analysis techniques are used in the planning game practice to define functionality of the system but primarily focuses on the client's requirements.
- The specific practice aims for an overall definition and grouping of requirements including functional and performance. The XP practice does not involve enough depth as required by this specific practice thus only partially satisfies the specific practice.

Specific Practice 3.3: Analyse Requirements
Practice Result: Satisfied

- The planning game practice fulfills this specific practice as the development team performs feasibility studies and prioritisation of the requirements thus analysing them before proceeding with development.

Specific Practice 3.4: Analyse Requirements to Achieve Balance
Practice Result: Partially Satisfied

- This specific practice deals with risk and by analysing the requirements it is checking whether the requirements are feasible to be included in the project.
- The planning game practice satisfies this specific practice as it uses proven models of prototyping. The development team do assess each requirement to check whether it is consistent, feasible, complete and realistic to the project.

Specific Practice 3.5: Validate Requirements
Practice Result: Satisfied

- Requirements validation is performed early in development with the client to gain confidence that the requirements are capable of guiding a development that results in successful final validation.
- The planning game practice satisfies this specific practice as the client is involved throughout the development. After each iteration the client can be used to validate the produced working software and client feedback can then be taken and integrated into the next iteration.

Mapping Key Points:

- The planning game practice involves the client throughout development.

- The use of requirement elicitation techniques such as use cases, storyboards, scenarios etc is common for both to understand the required functionality of the client.

- The development teams of the planning game practice perform feasibility studies to specify if the requirements are able to be fulfilled and that they are required. The overall aim of the analyse and validate requirements goal relates to certain key points of the planning game practice in that both are trying to justify and validate why that requirement is needed.

Requirements Management & The Planning Game

XP Practice – Planning Game
CMMI-Dev Process Area – Requirements Management

Table 30: Requirements Management & The Planning Game

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Planning Game Practice
SG 1 Manage Requirements SP 1.1 Obtain an Understanding of Requirements SP 1.2 Obtain Commitment to Requirements SP 1.3 Manage Requirements Changes SP 1.4 Maintain Bidirectional Traceability of Requirements SP 1.5 Identify Inconsistencies Between Project Work and Requirements	**(SG1)** Partially Satisfied	Client presents desired features through use cases / storyboards. Developers produce cost estimates + requirement prioritisation. Initial schedule / plan produced before commencing iteration. Running software produced every 2-4 weeks. Customer involved in evaluating software. Problems / Changes Analysed & Discussed Feedback taken and incorporated into next 2-4 week iteration.

Specific Goal 1: Manage Requirements

Goal Result: Partially Satisfied

Specific Practice 1.1: Obtain an Understanding of Requirements

Practice Result: Partially Satisfied

- This specific practice requires establishing criteria for distinguishing appropriate requirements providers and establishing objective criteria for the evaluation and acceptance of requirements.
- The planning game partially satisfies this specific practice as the evaluation and acceptance of requirements is provided by the client who validates the software after each iteration.
- Their feedback is then taken and integrated into the next cycle to be corrected or improved upon.
- The development team also analyse the requirements to prioritise them and check that they are valid which adds to understanding the requirements. However, criteria is supposed to be established in evaluating and accepting requirements, which is not established during the planning game practice thus partially satisfies this specific practice.

Mapping Key Points:

- Having the customer evaluate the software after an iteration provides acceptance that the requirements are correct. If not, their feedback is integrated into the next cycle and the requirement is improved then. New requirements can also be established.
- The development team analyse the requirements but do not use any criteria to prove that the requirements are correct.

Specific Practice 1.2: Obtain Commitment to Requirements

Practice Result: Satisfied

- This specific practice deals with agreements and commitments among those who have to carry out the activities to implement the requirements.
- The planning game practice satisfies this specific practice as the team set out an initial schedule / plan before each iteration commences, which will contain who has what tasks to complete.
- By documenting commitments in the schedule / plan before starting tasks, the planning game practice achieves the same aim as the specific practice.

Specific Practice 1.3: Manage Requirements Changes
Practice Result: Partially Satisfied

- This specific practice deals with documenting any changes that occur during development and allows change to be dealt with effectively.
- The planning game partially satisfies this practice as there is a review with the client after each 2-4 week iteration where requirement changes are dealt with at that point allowing the client to have an influence on decisions.
- The planning game practice does not fully document or keep records of the changes thus only partially satisfies the specific practice.

Specific Practice 1.4: Maintain Bi-directional Traceability of Requirements
Practice Result: Unsatisfied

- The planning game practice does not satisfy this specific practice as it fails to track requirements from beginning to the end of production.
- The client or developer will often make decisions that are not documented in the 2-4 week cycle thus cannot be traced through development to understand why they were made or what impact they have had.

Specific Practice 1.5: Identify Inconsistencies Between Project Work and Requirements
Practice Result: Unsatisfied

- There are no procedures in the planning game practice that address this specific practice as inconsistencies between project plans and requirements are not detected.
- The planning game practice focuses on encouraging development and not on analysing or producing documentation.

Design

Technical Solution & Design Improvement

XP Practice – Design Improvement
CMMI-Dev Process Area – Technical Solution

Table 31: Technical Solution & Design Improvement

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Design Improvement Practice
SG 1 Select Product Component Solutions SP 1.1 Develop Alternative Solutions and Selection Criteria SP 1.2 Select Product Component Solutions	**(SG1)** Unsatisfied	Continuing Simple Design
SG 2 Develop the Design SP 2.1 Design the Product or Product Component SP 2.2 Establish a Technical Data Package SP 2.3 Design Interfaces Using Criteria SP 2.4 Perform Make, Buy, or Reuse Analyses	**(SG2)** Unsatisfied	Communication Simplicity
SG 3 Implement the Product Design SP 3.1 Implement the Design SP 3.2 Develop Product Support Documentation	**(SG3)** Partially Satisfied	Clean Design Refactoring

Specific Goal 1: Select Product Component Solutions
Goal Result: Unsatisfied

Specific Practice 1.1 Develop Alternative Solutions and Selection Criteria
Practice Result: Unsatisfied

- This specific practice requires alternative design solutions to be identified and analysed to enable the selection of a balanced solution across the life of the product in terms of cost, schedule, and performance.
- The design improvement practice does not satisfy this specific practice as the design is continually developed upon throughout the project thus there is no need for a selection criterion to determine alternative solutions.

Specific Practice 1.2 Select Product Component Solutions
Practice Result: Unsatisfied

- The design improvement practice does not satisfy this specific practice as it produces detailed documentation regarding alternative solutions, evaluations and rationale. It also does not apply as the previous practice, 1.1, needs to be implemented for this specific practice to be satisfied.

Specific Goal 2: Develop the Design
Goal Result: Unsatisfied

Specific Practice 2.1 Design the Product or Product Component
Practice Result: Partially Satisfied

- This specific practice is partially satisfied by the design improvement practice as an initial design is mapped out but not to the extent that the specific practice requires such as architectural designs, design rules, and design modelling.
- Data schemas are generated, algorithms are developed, and heuristics are established to satisfy this specific practice, but the design improvement practice does not provide that amount of detail. It is more concerned with developing whilst designing, rather than producing large amounts of documentation.

Specific Practice 2.2 Establish a Technical Data Package
Practice Result: Unsatisfied

- A technical data package provides the developer with a comprehensive description of the product as it is developed.
- The design improvement practice cannot be applied to this specific practice as documentation and recording information about the product, as development progresses, is the main objective of this specific practice.

- The design improvement practice has no requirements regarding documenting how the product progresses over time.

Specific Practice 2.3 Design Interfaces Using Criteria
Practice Result: Unsatisfied

- The design improvement practice does not satisfy this specific practice as the design evolves throughout development. This specific practice requires the interfaces to be fully designed using criteria before development commences.

Specific Practice 2.4 Perform Make, Buy, or Reuse Analyses
Practice Result: Unsatisfied

- This specific practice refers to determining what products or product components will be acquired during the project.
- The design improvement practice does not satisfy this specific practice as it is concerned with how to keep the design simple by including techniques such as refactoring where as this specific practice refers to what will be contained in the design.

Specific Goal 3: Implement the Product Design
Goal Result: Unsatisfied

Specific Practice 3.1 Implement the Design
Practice Result: Partially Satisfied

- The design improvement practice uses refactoring techniques to remove unnecessary coding which can be applied to this specific practice of implementing the design.
- Although this specific practice focuses more on implementation, techniques such as refactoring can be used during the phase from design to implementation to improve the design.
- The specific practice requires data, services, processes and facilities etc to all be documented which the design improvement practice does not fully accomplish.

Specific Practice 3.2 Develop Product Support Documentation
Practice Result: Unsatisfied

- The design improvement practice does not satisfy this specific practice as it focuses on producing documents such as user manuals, online help guides, maintenance and operator manuals where as the design improvement practice is only concerned with the actual design of the product.

Technical Solution & Simple Design

XP Practice – Simple Design
CMMI-Dev Process Area – Technical Solution

Table 32:Technical Solution & Simple Design Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Simple Design Practice
SG 1 Select Product Component Solutions SP 1.1 Develop Alternative Solutions and Selection Criteria SP 1.2 Select Product Component Solutions **SG 2 Develop the Design** SP 2.1 Design the Product or Product Component SP 2.2 Establish a Technical Data Package SP 2.3 Design Interfaces Using Criteria SP 2.4 Perform Make, Buy, or Reuse Analyses **SG 3 Implement the Product Design** SP 3.1 Implement the Design SP 3.2 Develop Product Support Documentation	**(SG1)** Unsatisfied **(SG2)** Partially Satisfied **(SG3)** Partially Satisfied	Invest in Design Everyday Design Incrementally in Small, Safe Steps Design for That Day / Iteration Little Duplication Refactor Simplicity

Specific Goal 1: Select Product Component Solutions
Goal Result: Unsatisfied

Specific Practice 1.1 Develop Alternative Solutions and Selection Criteria
Practice Result: Unsatisfied

- This specific practice requires alternative design solutions to be identified and analysed to enable the selection of a balanced solution across the life of the product in terms of cost, schedule, and performance.
- The simple design practice does not satisfy this specific practice as it is concerned with designing only for a specific piece of functionality or design for that day where as the specific practice is focusing on a large amount of upfront design before coding.

Specific Practice 1.2 Select Product Component Solutions
Practice Result: Unsatisfied

- The simple design practice does not satisfy this specific practice as it produces detailed documentation regarding alternative solutions, evaluations and rationale.
- It also does not apply as the previous practice, 1.1, needs to be implemented for this specific practice to be satisfied.

Specific Goal 2: Develop the Design
Goal Result: Unsatisfied

Specific Practice 2.1 Design the Product or Product Component
Practice Result: Partially Satisfied

- This specific practice is partially satisfied by the simple design practice as an initial design is mapped out but not to the extent that the specific practice requires such as architectural designs, design rules, and design modelling.
- Data schemas are generated, algorithms are developed, and heuristics are established to satisfy this specific practice, but the simple design practice does not provide that amount of detail. It is more concerned with how to design, rather than producing large amounts of documentation.

Specific Practice 2.2 Establish a Technical Data Package
Practice Result: Unsatisfied

- A technical data package provides the developer with a comprehensive description of the product as it is developed. The simple design practice cannot be applied to this specific practice as documentation and recording information about the product, as development progresses, is the main objective of this specific practice. The simple design practice has no requirements regarding documenting how the product progresses over time.

Specific Practice 2.3 Design Interfaces Using Criteria
Practice Result: Unsatisfied

- The simple design practice does not satisfy this specific practice as too much detail is required to fulfil the specific practice with regards to interface design specifications, interface control documents, interface specification criteria and rationale for selected interface design.

Specific Practice 2.4 Perform Make, Buy, or Reuse Analyses
Practice Result: Unsatisfied

- This specific practice refers to determining what products or product components will be acquired during the project.
- The design improvement practice does not satisfy this specific practice as it is concerned with how to keep the design simple by including techniques such as refactoring where as this specific practice refers to what will be contained in the design.

Specific Goal 3: Implement the Product Design
Goal Result: Unsatisfied

Specific Practice 3.1 Implement the Design
Practice Result: Partially Satisfied

- The simple design practice suggests techniques to remove unnecessary coding which can be applied to this specific practice of implementing the design.
- Although this specific practice focuses more on implementation, techniques such as refactoring can be used during the phase from design to implementation to improve the design by keeping it simple.
- The specific practice requires data, services, processes and facilities etc to all be documented which the simple design practice does not accomplish.

Specific Practice 3.2 Develop Product Support Documentation
Practice Result: Unsatisfied

- The simple design practice does not satisfy this specific practice as it focuses on producing documents such as user manuals, online help guides, maintenance and operator manuals where as the simple design practice is only concerned with the actual design of the product and not the supporting documentation.

Implementation Activities

Product Integration & Pair Programming

XP Practice – Design Improvement
CMMI-Dev Process Area – Technical Solution

Table 33: Product Integration & Pair Programming Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Pair Programming Practice
SG 1 Prepare for Product Integration SP 1.1 Determine Integration Sequence SP 1.2 Establish the Product Integration Environment SP 1.3 Establish Product Integration Procedures and Criteria	**(SG1)** Unsatisfied	Keep Each Other On Task Clarify Ideas
SG 2 Ensure Interface Compatibility SP 2.1 Review Interface Descriptions for Completeness SP 2.2 Manage Interfaces	**(SG2)** Unsatisfied	Produce Software
SG 3 Assemble Product Components and Deliver the Product SP 3.1 Confirm Readiness of Product Components for Integration SP 3.2 Assemble Product Components SP 3.3 Evaluate Assembled Product Components SP 3.4 Package and Deliver the Product or Product Component	**(SG3)** Partially Satisfied	Brainstorm Developer Accountability Pair Support

Specific Goal 1: Prepare for Product Integration
Goal Result: Unsatisfied

Specific Practice 1.1 Determine Integration Sequence
Practice Result: Unsatisfied

- This specific practice produces an integration sequence that provides for incremental assembly and evaluation of product components that should provide a problem-free foundation for incorporation of other product components as they become available.
- The pair programming practice does not satisfy this specific practice as it is concerned with the physical coding of the software product rather than the integrating of available components to the overall system.

Specific Practice 1.2 Establish the Product Integration Environment
Practice Result: Unsatisfied

- This specific practice establishes and maintains the environment needed to support the integration of the product components. This could be the decision to buy an off the shelf development environment or develop, as new, an integrated development environment to build the software.
- The pair programming practice cannot be applied to this specific practice.

Specific Practice 1.3 Establish Product Integration Procedures and Criteria
Practice Result: Unsatisfied

- The procedures and criteria established by this specific practice define how the product components are to be verified, functions they are expected to have, and when the product is fully assembled, how it should be validated and delivered.
- The pair programming practice cannot be applied to this specific practice as the specific practice's objective is to establish procedures and criterion.

Specific Goal 2: Ensure Interface Compatibility
Goal Result: Unsatisfied

Specific Practice 2.1 Review Interface Descriptions for Completeness
Practice Result: Unsatisfied

- The pair programming practice does not satisfy this specific practice as it deals with managing interface requirements, specifications, and designs to help ensure that implemented interfaces will be complete and compatible.

Specific Practice 2.2 Manage Interfaces
Practice Result: Unsatisfied

- This specific practice manages interfaces including maintenance of the consistency of the interfaces throughout the life of the system.
- The pair programming practice does not satisfy this specific practice as it is concerned with managing the interfaces where as the pair programming practice relates to physical coding and problem solving.

Specific Goal 3: Assemble Product Components and Deliver the Product
Goal Result: Partially Satisfied

Specific Practice 3.1 Confirm Readiness of Product Components for Integration
Practice Result: Unsatisfied

- The purpose of this specific practice is to ensure that the properly identified product component meets its description and can actually be assembled according to the product integration sequence and available procedures.
- It is difficult to apply the pair programming practice to this specific practice as it does not delve into same amount of detail, which the specific practice requires to be fulfilled.

Specific Practice 3.2 Assemble Product Components
Practice Result: Satisfied

- The pair programming practice satisfies this specific practice as both are trying to achieve the same goal of building the components of the system.

Specific Practice 3.3 Evaluate Assembled Product Components
Practice Result: Unsatisfied

- This evaluation involves examining and testing assembled product components for performance, suitability, and readiness using the available procedures and environment.
- This specific practice does not relate fully to implementation, the focus is more on testing thus the pair programming practice does not satisfy this specific practice.

Specific Practice 3.4 Package and Deliver the Product or Product Component
Practice Result: Unsatisfied

- This specific practice cannot be satisfied by the pair programming practice as they represent different activities. The specific practice refers to how the final system is packaged and installed when fully complete.

Validation & Verification

Validation & Client Tests

XP Practice – Client Tests
CMMI-Dev Process Area – Validation

Table 34: Validation & Client Tests Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Client Tests Practice
SG 1 Prepare for Validation SP 1.1 Select Products for Validation SP 1.2 Establish the Validation Environment SP 1.3 Establish Validation Procedures and Criteria **SG 2 Validate Product or Product Components** SP 2.1 Perform Validation SP 2.2 Analyse Validation Results	**(SG1)** Partially Satisfied **(SG2)** Satisfied	Developer Knows What to Test Client Involved Client Expectations Understood Checks Conformation With Requirements Client Tests Analysed

Specific Goal 1: Prepare for Validation

Goal Result: Partially Satisfied

Specific Practice 1.1 Select Products for Validation

Practice Result: Satisfied

- Products and product components are selected for validation on the basis of their relationship to the client needs.
- The client tests practice satisfies this specific practice as the needs of the client are being tested due to the client preparing the tests.

Specific Practice 1.2 Establish the Validation Environment

Practice Result: Satisfied

- The client test practice satisfies this specific practice as the environment for testing is established which may include test tools, temporary embedded test software and recording tools for analysis.
- The important factor is that the practice will have a validation environment established to perform the client tests thus satisfying this specific practice.

Specific Practice 1.3 Establish Validation Procedures and Criteria

Practice Result: Partially Satisfied

- Validation procedures and criteria are defined to ensure that the product or product component will fulfill its intended use when placed in its intended environment.
- The client tests partially satisfy this specific practice as client acceptance test cases involved in the client tests practices, meet the need of validation procedures as stated by this specific practice.
- Documentation such as performance thresholds, standards and environmental performance will not be produced by the client tests thus the specific practice is only partially satisfied.

Specific Goal 2: Validate Product or Product Components

Goal Result: Satisfied

Specific Practice 2.1 Perform Validation

Practice Result: Satisfied

- Client tests satisfy this specific practice as validation results are obtained where they can be analysed to check whether they achieve the client's needs. Documentation produced that is common to both is validation reports and results, run logs and client demonstrations.

Specific Practice 2.2 Analyse Validation Results
Practice Result: Satisfied

- The client tests practice satisfies this specific practice as the results are analysed to reveal if the client's needs are met.
- As the tests involve the client, feedback is given that reports whether the tests are successful or fail, and will reveal a probable cause of failure. The client tests satisfy this specific practice as the results are collected in both practices and determine whether to proceed with building further requirements on the next iteration or to correct any problems that the tests have detected.

Validation & Small Releases

XP Practice – Small Releases
CMMI-Dev Process Area – Validation

Table 35: Validation & Small Releases Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Small Releases Practice
SG 1 Prepare for Validation SP 1.1 Select Products for Validation SP 1.2 Establish the Validation Environment SP 1.3 Establish Validation Procedures and Criteria **SG 2 Validate Product or Product Components** SP 2.1 Perform Validation SP 2.2 Analyse Validation Results	**(SG1)** Partially Satisfied **(SG2)** Satisfied	Working, Tested Software Produced Frequently Highest Prioritised Requirements Completed First Progress Detected Testing Completed In Every Release Client Can Contribute Feedback After Each Release Results Analysed

Specific Goal 1: Prepare for Validation

Goal Result: Partially Satisfied

Specific Practice 1.1 Select Products for Validation

Practice Result: Satisfied

- Products and product components are selected for validation on the basis of their relationship to the client needs.
- The small release practice satisfies this specific practice as early testing focuses on the developed software of that iteration thus they are selected for validation.
- As the client is involved in testing, the testing can be used to determine whether the software satisfies their needs.

Specific Practice 1.2 Establish the Validation Environment

Practice Result: Satisfied

- The client test practice satisfies this specific practice as the environment for testing is already established as part of the small release practice.
- Without having the environment to test, the small release practice would be less effective. The important factor is that the small release practice will have a validation environment established to perform the client tests thus satisfying this specific practice.

Specific Practice 1.3 Establish Validation Procedures and Criteria

Practice Result: Unsatisfied

- Validation procedures and criteria are defined to ensure that the product or product component will fulfill its intended use when placed in its intended environment.
- The small release practice does not satisfy this specific practice as it focuses on frequent releases and building tested software in small stages.
- This specific practice requires a testing criteria and procedures to be documented such as performance thresholds, standards, and environmental performance which are not produced by the small releases practice.

Specific Goal 2: Validate Product or Product Components

Goal Result: Satisfied

Specific Practice 2.1 Perform Validation

Practice Result: Satisfied

- The small release practice satisfies this specific practice as validation results are obtained after the tests of a small release are performed where they can be analysed to check whether they achieve the client's needs.
- Documentation produced that is common to both is validation reports and results, run logs and client demonstrations.

Specific Practice 2.2 Analyse Validation Results
Practice Result: Satisfied

- The small release practice satisfies this specific practice as the results of the tests are analysed to reveal if the client's needs are met.
- As the tests in the small release involve the client, feedback is given that reports whether the tests are successful.
- The client tests involved in each small release satisfy this specific practice as the results are collected in both practices and determine whether to proceed with building further requirements on the next iteration or to correct any problems that the tests have detected.

Validation & Pair Programming

XP Practice – Pair Programming
CMMI-Dev Process Area – Validation

Table 36: Validation & Pair Programming Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Pair Programming Practice
SG 1 Prepare for Validation SP 1.1 Select Products for Validation SP 1.2 Establish the Validation Environment SP 1.3 Establish Validation Procedures and Criteria **SG 2 Validate Product or Product Components** SP 2.1 Perform Validation SP 2.2 Analyse Validation Results	**(SG1)** Partially Satisfied **(SG2)** Partially Satisfied	Keep Each Other On Task Clarify Ideas Produce Software Brainstorm Developer Accountability Pair Support

Specific Goal 1: Prepare for Validation
Goal Result: Partially Satisfied

Specific Practice 1.1 Select Products for Validation
Practice Result: Partially Satisfied

- Products and product components are selected for validation on the basis of their relationship to the client needs. The developers in the pair programming practice, who code the software, select what should be tested and are the most suitable on the team to make this decision as it is their code that is being validated.

Specific Practice 1.2 Establish the Validation Environment
Practice Result: Unsatisfied

- The pair programming practice does not satisfy this specific practice as no testing is performed during the pair programming. The practice focuses on coding of the design.

Specific Practice 1.3 Establish Validation Procedures and Criteria
Practice Result: Unsatisfied

- Validation procedures and criteria are defined to ensure that the product or product component will fulfill its intended use when placed in its intended environment.
- The pair programming practice does not satisfy this specific practice as it focuses on coding rather than establishing procedures and criteria.

Specific Goal 2: Validate Product or Product Components
Goal Result: Partially Satisfied

Specific Practice 2.1 Perform Validation
Practice Result: Partially Satisfied

- The pair programming practice satisfies this specific practice as validation is performed continuously by working in pairs.
- By programming in pairs, any coding that is performed has at least been checked by the two initial developers. Throughout the implementation, as one codes the other analyses his partners code for errors.

Specific Practice 2.2 Analyse Validation Results
Practice Result: Unsatisfied

- The pair programming practice does not satisfy this specific practice as no errors are detected when coding or analysed or documented.

Validation & Test Driven Development

XP Practice – Test Driven Development
CMMI-Dev Process Area – Validation

Table 37: Validation & Test Driven Mappings

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Test Driven Development Practice
SG 1 Prepare for Validation SP 1.1 Select Products for Validation SP 1.2 Establish the Validation Environment SP 1.3 Establish Validation Procedures and Criteria **SG 2 Validate Product or Product Components** SP 2.1 Perform Validation SP 2.2 Analyse Validation Results	**(SG1)** Partially Satisfied **(SG2)** Satisfied	Testing Before Coding Early Feedback Received Focus On Client Needs by Testing First Progress Shown Errors Detected Early

Specific Goal 1: Prepare for Validation

Goal Result: Partially Satisfied

Specific Practice 1.1 Select Products for Validation

Practice Result: Satisfied

- Products and product components are selected for validation on the basis of their relationship to the client needs. The test driven development satisfies this specific practice by focusing on the clients' needs from the beginning of development as they are forced to test first.
- This allows products to be selected based on an understanding of the client requirements and needs.

Specific Practice 1.2 Establish the Validation Environment

Practice Result: Satisfied

- The test driven development practice satisfies this specific practice as testing is the main objective of the XP practice.
- Without the testing environment the practice would be useless thus an established validation environment is setup.

Specific Practice 1.3 Establish Validation Procedures and Criteria

Practice Result: Unsatisfied

- Validation procedures and criteria are defined to ensure that the product or product component will fulfill its intended use when placed in its intended environment.
- The test driven development practice does not satisfy this specific practice as it focuses on testing and coding rather than establishing procedures and criteria.

Specific Goal 2: Validate Product or Product Components

Goal Result: Satisfied

Specific Practice 2.1 Perform Validation

Practice Result: Satisfied

- Validation is performed throughout the test driven practice as the client's needs are focused upon during testing and then coded. Any requirements that are unclear can be clarified by the client to allow the developer to gain a thorough understanding of the client's needs.
- This allows the product or piece of functionality to demonstrate that it fulfills its intended use.

Specific Practice 2.2 Analyse Validation Results
Practice Result: Satisfied

- Test driven development produces test results that can be analysed by the development team and the client.
- The test driven development practice produces results which acts as a safety precaution to the developer as they know that a change they have made to the code has passed or failed due to the test

Verification & Client Tests

XP Practice – Client Tests
CMMI-Dev Process Area – Verification

Table 38: Verification & Client Tests Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Client Tests Practice
SG 1 Prepare for Verification SP 1.1 Select Work Products for Verification SP 1.2 Establish the Verification Environment SP 1.3 Establish Verification Procedures and Criteria	**(SG1)** Partially Satisfied	Developer Knows What to Test Client Involved
SG 2 Perform Peer Reviews SP 2.1 Prepare for Peer Reviews SP 2.2 Conduct Peer Reviews SP 2.3 Analyse Peer Review Data	**(SG2)** Unsatisfied	Client Expectations Understood
SG 3 Verify Selected Work Products SP 3.1 Perform Verification SP 3.2 Analyse Verification Results	**(SG3)** Satisfied	Checks Conformation With Requirements Client Tests Analysed

Specific Goal 1: Prepare for Verification

Goal Result: Partially Satisfied

Specific Practice 1.1 Select Work Products for Verification
Practice Result: Satisfied

- Work products are selected based on their contribution to meeting project objectives and project requirements.
- The client tests satisfy this specific practice by focusing on the clients' requirements from the beginning of development by being forced to test first. This allows work products to be selected based on an understanding of the client, their requirements, and their needs.
- Methods that are common to both specific practice and the client tests practice are test cases and acceptance tests.

Specific Practice 1.2 Establish the Verification Environment
Practice Result: Satisfied

- The client test practice satisfies this specific practice as the environment for testing is established which may include test tools, temporary embedded test software and recording tools for analysis.
- The important factor is that the practice will have a validation environment established to perform the client tests thus satisfying this specific practice.

Specific Practice 1.3 Establish Verification Procedures and Criteria
Practice Result: Unsatisfied

- Verification criteria are defined to ensure that the work products meet their requirements.
- The client test practice does not satisfy this specific practice as documents that are likely to be developed are standards, supplier agreements, and policies all of which are not related to the client tests practice.

Specific Goal 2 Perform Peer Reviews

Goal Result: Unsatisfied

Peer reviews involve a methodical examination of work products by the producers' peers to identify defects for removal and to recommend other changes that are needed.

Specific Practice 2.1 Prepare for Peer Reviews
Practice Result: Unsatisfied

- No peer reviews are carried out by the client tests practice.

Specific Practice 2.2 Conduct Peer Reviews
Practice Result: Unsatisfied

- No peer reviews are carried out by the client tests practice.

Specific Practice 2.3 Analyse Peer Review Data
Practice Result: Unsatisfied

- No peer reviews are carried out by the client tests practice.

Specific Goal 3: Verify Selected Work Product
Goal Result: Satisfied

Specific Practice 3.1 Perform Verification
Practice Result: Satisfied

- Client tests satisfy this specific practice as validation results are obtained where they can be analysed to check whether they achieve the project requirements and objectives.
- Documentation produced that is common to both is validation reports and results, run logs and client demonstrations.

Specific Practice 3.2 Analyse Verification Results
Practice Result: Satisfied

- The client tests practice satisfies this specific practice as the results are analysed to reveal if the client's needs are met.
- As the tests involve the client, feedback is given that reports whether the tests are successful or fail, and will reveal a probable cause of failure.
- The client tests satisfy this specific practice as the results are collected in both practices and determine whether to proceed with building further requirements on the next iteration or to correct any problems that the tests have detected.

Verification & Small Releases

XP Practice – Small Releases
CMMI-Dev Process Area – Verification

Table 39: Verification & Small Releases Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Small Releases Practice
SG 1 Prepare for Verification SP 1.1 Select Work Products for Verification SP 1.2 Establish the Verification Environment SP 1.3 Establish Verification Procedures and Criteria **SG 2 Perform Peer Reviews** SP 2.1 Prepare for Peer Reviews SP 2.2 Conduct Peer Reviews SP 2.3 Analyse Peer Review Data **SG 3 Verify Selected Work Products** SP 3.1 Perform Verification SP 3.2 Analyse Verification Results	**(SG1)** Partially Satisfied **(SG2)** Unsatisfied **(SG3)** Satisfied	Working, Tested Software Produced Frequently Highest Prioritised Requirements Completed First Progress Detected Testing Completed In Every Release Client Can Contribute Feedback After Each Release Results Analysed

Specific Goal 1: Prepare for Verification

Goal Result: Partially Satisfied

Specific Practice 1.1 Select Work Products for Verification
Practice Result: Satisfied

- Work products are selected based on their contribution to meeting project objectives and project requirements.
- The small release practice satisfies this specific practice as early testing focuses on the developed software of that iteration thus they are selected for verification.
- This allows work products to be selected based on an understanding of the client, their requirements, and their needs.

Specific Practice 1.2 Establish the Verification Environment
Practice Result: Satisfied

- The client test practice satisfies this specific practice as the environment for testing is already established as part of the small release practice.
- Without having the environment to test, the small release practice would be less effective.
- The important factor is that the release practice sets up a validation environment to perform the client tests thus satisfying this specific practice.

Specific Practice 1.3 Establish Verification Procedures and Criteria
Practice Result: Unsatisfied

- Verification criteria are defined to ensure that the work products meet their requirements.
- The small release practice does not satisfy this specific practice as documents that are likely to be developed are standards, supplier agreements, and policies all of which are not produced by the small release practice.

Specific Goal 2 Perform Peer Reviews

Goal Result: Unsatisfied

- Peer reviews involve a methodical examination of work products by the producers' peers to identify defects for removal and to recommend other changes that are needed.

Specific Practice 2.1 Prepare for Peer Reviews
Practice Result: Unsatisfied

- No peer reviews are carried out by the small release practice.

Specific Practice 2.2 Conduct Peer Reviews
Practice Result: Unsatisfied

- No peer reviews are carried out by the small release practice.

Specific Practice 2.3 Analyse Peer Review Data
Practice Result: Unsatisfied

- No peer reviews are carried out by the small release practice.

Specific Goal 3: Verify Selected Work Product
Goal Result: Satisfied

Specific Practice 3.1 Perform Verification
Practice Result: Satisfied

- The small release practice satisfies this specific practice as validation results are obtained after the tests of a small release are performed where they can be analysed to check whether they achieve the client's needs.
- Documentation produced that is common to both is verification reports and results, run logs and client demonstrations.

Specific Practice 3.2 Analyse Verification Results
Practice Result: Satisfied

- The small release practice satisfies this specific practice as the results of the tests are analysed to reveal if the client's needs are met.
- As the tests in the small release involve the client, feedback is given that reports whether the tests are successful.
- The client tests involved in each small release satisfy this specific practice as the results are collected in both practices and determine whether to proceed with building further requirements on the next iteration or to correct any problems that the tests have detected.

Verification & Pair Programming

XP Practice – Pair Programming
CMMI-Dev Process Area – Verification

Table 40: Verification & Pair Programming Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Pair Programming Practice
SG 1 Prepare for Verification SP 1.1 Select Work Products for Verification SP 1.2 Establish the Verification Environment SP 1.3 Establish Verification Procedures and Criteria **SG 2 Perform Peer Reviews** SP 2.1 Prepare for Peer Reviews SP 2.2 Conduct Peer Reviews SP 2.3 Analyse Peer Review Data **SG 3 Verify Selected Work Products** SP 3.1 Perform Verification SP 3.2 Analyse Verification Results	**(SG1)** Partially Satisfied **(SG2)** Partially Satisfied **(SG3)** Satisfied	Developers Focus Each Other On The Task Clarify Ideas Produce Software Brainstorm Developer Accountability Pair Support

Specific Goal 1: Prepare for Verification

Goal Result: Partially Satisfied

Specific Practice 1.1 Select Work Products for Verification

Practice Result: Partially Satisfied

- Work products are selected based on their contribution to meeting project objectives and project requirements.
- The developers in the pair programming practice, who code the software, select what should be tested and are the most suitable on the team to make that decision as it is their code that is being verified.
- Testing has a large contribution to verification and is not covered by the pair programming practice thus it only partially satisfies this specific practice.

Specific Practice 1.2 Establish the Verification Environment

Practice Result: Unsatisfied

- The pair programming practice does not satisfy this specific practice as no testing is performed during the pair programming. The practice focuses on coding of the design.

Specific Practice 1.3 Establish Verification Procedures and Criteria

Practice Result: Unsatisfied

- Verification criteria are defined to ensure that the work products meet their requirements.
- The pair programming practice does not satisfy this specific practice as it focuses on coding rather than establishing procedures and criteria.

Specific Goal 2 Perform Peer Reviews

Goal Result: Partially Satisfied

- Peer reviews involve a methodical examination of work products by the producers' peers to identify defects for removal and to recommend other changes that are needed.

Specific Practice 2.1 Prepare for Peer Reviews

Practice Result: Unsatisfied

- The pair programming practice does not satisfy this specific practice as no preparation is conducted for the peer review as it is a continuous process in the pair programming practice.

Specific Practice 2.2 Conduct Peer Reviews
Practice Result: Satisfied

- As the programming occurs in pairs, peer reviewing is conducted continually by the pair programming practice.
- One of the purposes of conducting a peer review is to find and remove defects early, which is one reason for having the pair programming structure. One can develop and the other checks the code to detect errors.

Specific Practice 2.3 Analyse Peer Review Data
Practice Result: Unsatisfied

- The pair programming practice does not satisfy this specific practice as the data is not stored for future reference or analysed to discover why the error occurred.

Specific Goal 3: Verify Selected Work Product
Goal Result: Satisfied

Specific Practice 3.1 Perform Verification
Practice Result: Partially Satisfied

- The pair programming practice partially satisfies this specific practice as validation is performed continuously by working in pairs.
- By programming in pairs, any coding that is performed has at least been checked by the two initial developers.
- Throughout the implementation, as one codes the other analyses his partners code for errors. To fully satisfy the specific practice, the pair programming practice would have to record results, have run logs, and produce reports which it does not carry out.

Specific Practice 3.2 Analyse Verification Results
Practice Result: Unsatisfied

- The pair programming practice does not satisfy this specific practice as no errors detected when coding are analysed or documented.

Verification & Test Driven Development

XP Practice – Test Driven Development
CMMI-Dev Process Area – Verification

Table 41: Verification & Test Driven Development Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Test Driven Development Practice
SG 1 Prepare for Verification SP 1.1 Select Work Products for Verification SP 1.2 Establish the Verification Environment SP 1.3 Establish Verification Procedures and Criteria **SG 2 Perform Peer Reviews** SP 2.1 Prepare for Peer Reviews SP 2.2 Conduct Peer Reviews SP 2.3 Analyse Peer Review Data **SG 3 Verify Selected Work Products** SP 3.1 Perform Verification SP 3.2 Analyse Verification Results	**(SG1)** Partially Satisfied **(SG2)** Unsatisfied **(SG3)** Satisfied	Testing Before Coding Early Feedback Received Focus On Client Needs by Testing First Progress Shown Errors Detected Early

Specific Goal 1: Prepare for Verification

Goal Result: Partially Satisfied

Specific Practice 1.1 Select Work Products for Verification
Practice Result: Satisfied

- Work products are selected based on their contribution to meeting project objectives and project requirements. The test driven development satisfies this specific practice by focusing on the clients' needs from the beginning of development as they are forced to test first.
- This allows products to be selected based on an understanding of the client requirements and needs.
- Testing makes a large contribution to verification and covered by the test driven development practice producing test results and recordings satisfies this specific practice.

Specific Practice 1.2 Establish the Verification Environment
Practice Result: Satisfied

- The test driven development practice satisfies this specific practice as testing is the main objective of the XP practice.
- Without the testing environment the practice would be useless thus an established validation environment is setup.

Specific Practice 1.3 Establish Verification Procedures and Criteria
Practice Result: Unsatisfied

Verification criteria are defined to ensure that the work products meet their requirements. The test driven development practice does not satisfy this specific practice as it focuses on testing and coding rather than establishing procedures and criteria.

Specific Goal 2 Perform Peer Reviews

Goal Result: Unsatisfied

- Peer reviews involve a methodical examination of work products by the producers' peers to identify defects for removal and to recommend other changes that are needed.

Specific Practice 2.1 Prepare for Peer Reviews
Practice Result: Unsatisfied

- No peer reviews are carried out by the test driven development practice.

Specific Practice 2.2 Conduct Peer Reviews
Practice Result: Unsatisfied

- No peer reviews are carried out by the test driven development practice.

Specific Practice 2.3 Analyse Peer Review Data
Practice Result: Unsatisfied

- No peer reviews are carried out by the test driven development practice.

Specific Goal 3: Verify Selected Work Product
Goal Result: Satisfied

Specific Practice 3.1 Perform Verification
Practice Result: Satisfied

- Verification is performed throughout the test driven practice as the client's needs are focused upon during testing and then coded.
- Any requirements that are unclear can be clarified by the client to allow the developer to gain a thorough understanding of the client's needs.
- This allows the product or piece of functionality to demonstrate that it fulfills the project requirements and objectives.
- To satisfy the specific practice, the test driven development practice records results, produces run logs and reports to be analysed.

Specific Practice 3.2 Analyse Verification Results
Practice Result: Satisfied

- The test driven development practice produces test results that can be analysed by the development team and the client.
- The test driven development practice produces results, which acts as a safety precaution to the developer as they know that a change they have made to the code has passed or failed due to the test.

Software Project Management

Casual Analysis & Resolution & The Planning Game

XP Practice – The Planning Game
CMMI-Dev Process Area – Casual Analysis & Resolution

Table 42: Casual Analysis & Resolution & The Planning Game Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of The Planning Game Practice
SG 1 Determine Causes of Defects SP 1.1 Select Defect Data for Analysis SP 1.2 Analyse Causes **SG 2 Address Causes of Defects** SP 2.1 Implement the Action Proposals SP 2.2 Evaluate the Effect of Changes SP 2.3 Record Data	**(SG1)** Partially Satisfied **(SG2)** Partially Satisfied	Client presents desired features through use cases / storyboards. Developers produce cost estimates + requirement prioritisation. Initial schedule / plan produced before commencing iteration. Running software produced every 2-4 weeks. Customer involved in evaluating software. Problems / Changes Analysed & Discussed Feedback taken and incorporated into next 2-4 week iteration.

Specific Goal 1: Determine Causes of Defects

Goal Result: Partially Satisfied

Specific Practice 1.1 Select Defect Data for Analysis

Practice Result: Satisfied

- The planning game satisfies this specific practice as the development team and client come together after each 2-4 week iteration and discuss any problems found during the iteration.
- The defects may come from requirements analysis, testing, or even a defect discovered by the client. At these meetings, defects can be highlighted to determine the cost, time, and resources needed to fix them and consideration can be given to how they may impact the project.

Specific Practice 1.2 Analyse Causes

Practice Result: Partially Satisfied

- At the meetings, the defects discovered during evaluation can be added to an action proposal that determines how they shall be resolved.
- As the whole team are present at these planning game meetings, those who have an understanding of the defect are typically responsible for performing the task to resolve the problem and plans can be arranged for this at the meeting.

Specific Goal 2: Address Causes of Defects

Goal Result: Partially Satisfied

Specific Practice 2.1 Implement the Action Proposals

Practice Result: Partially Satisfied

- The planning game practice partially satisfies this specific practice as the person chosen to resolve the defect, implements the plans discussed during the meeting.
- The planning game practice partially satisfies this specific practice as it will focus on fixing the defect during the evaluation and then continue with development, but the specific practice requires a large amount of supporting documentation which will not be produced.

Specific Practice 2.2 Evaluate the Effect of Changes

Practice Result: Partially Satisfied

- The planning game practice partially satisfies this specific practice as it will not monitor the effect of resolving the defect.
- If there are any problems that develop due to the resolved defect then they will be discovered during the next iteration of the evaluation phase where they can be brought to attention at the next team meeting.

Specific Practice 2.3 Record Data
Practice Result: Unsatisfied

- Data on the defect would not be recorded by the planning game practice. The information recorded would be used across the organisation or within similar projects. Data recorded would be costs, rationale on decisions, defect data, use of resources etc.

Integrated Project Management & The Planning Game

XP Practice – The Planning Game
CMMI-Dev Process Area – Integrated Project Management

Table 43: Integrated Project Management & The Planning Game Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of The Planning Game Practice
SG 1 Use the Project's Defined Process SP 1.1 Establish the Project's Defined Process SP 1.2 Use Organizational Process Assets for Planning Project Activities SP 1.3 Establish the Project's Work Environment SP 1.4 Integrate Plans SP 1.5 Manage the Project Using the Integrated Plans SP 1.6 Contribute to the Organizational Process Assets **SG 2 Coordinate and Collaborate with Relevant Stakeholders** SP 2.1 Manage Stakeholder Involvement SP 2.2 Manage Dependencies SP 2.3 Resolve Coordination Issues	**(SG1)** Partially Satisfied **(SG2)** Partially Satisfied	Client presents desired features through use cases / storyboards. Developers produce cost estimates + requirement prioritisation. Initial schedule / plan produced before commencing iteration. Running software produced every 2-4 weeks. Customer involved in evaluating software. Problems / Changes Analysed & Discussed Feedback taken and incorporated into next 2-4 week iteration.

Specific Goal 1 Use the Project's Defined Process

Goal Result: Partially Satisfied

Specific Practice 1.1 Establish the Project's Defined Process
Practice Result: Satisfied

- The planning game practice sets out activities that occur throughout the project's lifecycle and help staff and stakeholders establish initial requirements and plans. This specific goal is satisfied by the planning game practice.

Specific Practice 1.2 Use Organisational Process Assets for Planning Project Activities
Practice Result: Satisfied

- Although organisational process assets differ from organisation to organisation, the planning game practice does set out a schedule for completion of activities during the project.
- Estimates of time, cost and resources are used as a basis to plan project activities, which is the main objective of this specific practice.

Specific Practice 1.3 Establish the Project's Work Environment
Practice Result: Partially Satisfied

- The planning game practice partially satisfies this specific practice by setting out equipment and tools to complete the project.
- The level of detail required to fully satisfy this specific practice produces a large amount of documentation that is not produced by the XP practice including user surveys and results of the tools, usage, performance, and maintenance records and include support services required to use the tools.

Specific Practice 1.4 Integrate Plans
Practice Result: Unsatisfied

- This specific practice requires the project plan to be integrated with other plans that may affect the project plan such as software quality assurance plans, configuration management plans, and other documents. The planning game practice does not satisfy this specific practice.

Specific Practice 1.5 Manage the Project Using the Integrated Plans
Practice Result: Partially Satisfied

- The planning game practice uses the initial plan / schedule to manage the project during the iteration, which is adapted before the next iteration to accommodate change thus partially satisfying this specific practice.
- The specific practice requires progress reports, revised requirements, plans, and commitments to be recorded which the planning game practice does not satisfy.

- The planning game practice creates a new schedule / plan every iteration, where requirements or activities that need to be altered are discussed and recorded at this stage.

Specific Practice 1.6 Contribute to the Organisational Process Assets
Practice Result: Unsatisfied

- The planning game practice does not satisfy this specific practice as it aims to record details on the current project that could be used in future projects.
- Documentation produced by this specific practice regards improvements that could be made to the project, processes and product measures collected and lessons learned etc. The planning game practice does not record information for future projects.

Specific Goal 2 Coordinate and Collaborate with Relevant Stakeholders
Goal Result: Partially Satisfied

Specific Practice 2.1 Manage Stakeholder Involvement
Practice Result: Satisfied

- The planning game practice satisfies this specific practice as stakeholder involvement is managed by having scheduled meetings after each iteration that involves all stakeholders of the project.
- This is where stakeholder issues can be raised, project progress discussed and recommendations for resolving any stakeholder problems put forward thus satisfying this specific practice.

Specific Practice 2.2 Manage Dependencies
Practice Result: Partially Satisfied

- The planning game practice partially satisfies this specific practice by having an evaluation review of the software developed during the iteration. This is performed with all stakeholders of the system.
- The tracking of dependencies and defects is not covered by the planning game practice as well as the amount of documentation required to fulfill the specific practice.

Specific Practice 2.3 Resolve Coordination Issues
Practice Result: Satisfied

- The planning game practice partially satisfies this specific practice by having an evaluation review of the software developed during the iteration. This is performed with all stakeholders of the system and is where co-ordination issues would be resolved.

Measurement and Analysis & The Planning Game

XP Practice – The Planning Game
CMMI-Dev Process Area – Measurement & Analysis

Table 44: Measurement and Analysis & The Planning Game Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of The Planning Game Practice
SG 1 Align Measurement and Analysis Activities SP 1.1 Establish Measurement Objectives SP 1.2 Specify Measures SP 1.3 Specify Data Collection and Storage Procedures SP 1.4 Specify Analysis Procedures	**(SG1)** Partially Satisfied	Client presents desired features through use cases / storyboards. Developers produce cost estimates + requirement prioritisation. Initial schedule / plan produced before commencing iteration.
SG 2 Provide Measurement Results SP 2.1 Collect Measurement Data SP 2.2 Analyze Measurement Data SP 2.3 Store Data and Results SP 2.4 Communicate Results	**(SG2)** Partially Satisfied	Running software produced every 2-4 weeks. Customer involved in evaluating software. Problems / Changes Analysed & Discussed Feedback taken and incorporated into next 2-4 week iteration.

Specific Goal 1 Align Measurement and Analysis Activities

Goal Result: Partially Satisfied

Specific Practice 1.1 Establish Measurement Objectives
Practice Result: Partially Satisfied

- The planning game practice partially satisfies this specific practice as it fulfills establishing measurement objectives by allowing the client and development team to plan objectives prior to the start of coding.
- The planning game practice is only focused on achieving the measurement objective and then the objective is disregarded, where as this specific practice has reviews and documentation produced to enable traceability throughout the project.
- Example measurement objectives could be to reduce time to delivery, reduce total lifecycle cost, deliver specified functionality completely, improve prior quality levels etc.

Specific Practice 1.2 Specify Measures
Practice Result: Unsatisfied

- The planning game practice does not satisfy this specific practice as it sets up how the objectives will be achieved. I.e. via effort and cost, or work product size such as lines of code.
- The planning game practice does not set out any measurements to achieve the objectives established.
- No other XP practice could be used to satisfy this specific practice.

Specific Practice 1.3 Specify Data Collection and Storage Procedures
Practice Result: Unsatisfied

- This specific practice is unsatisfied by the planning game practice as no mechanisms are in place to specify how to collect data or to store data that could be used for further use.
- XP is applied in a project to project basis, one project at a time, where as this CMMI-Dev framework is supposed to be used in every project in the same rigorous manner regardless of project variation.
- No other XP practice could be used to satisfy this specific practice.

Specific Practice 1.4 Specify Analysis Procedures
Practice Result: Unsatisfied

- This specific practice is unsatisfied by the planning game practice as no analysis procedures are conducted within XP.
- There seems to be no form of measuring any objectives set in the planning stage by XP.
- No other XP practice could be used to satisfy this specific practice.

Specific Goal 2 Provide Measurement Results

Goal Result: Unsatisfied

Specific Practice 2.1 Collect Measurement Data

Practice Result: Unsatisfied

- This specific practice is unsatisfied by the planning game practice as no data in XP is collected to be analysed.
- No other XP practice could be used to satisfy this specific practice.

Specific Practice 2.2 Analyse Measurement Data

Practice Result: Unsatisfied

- This specific practice is unsatisfied by the planning game practice as no data in XP is collected to be analysed.
- No other XP practice could be used to satisfy this specific practice.

Specific Practice 2.3 Store Data and Results

Practice Result: Unsatisfied

- This specific practice is unsatisfied by the planning game practice as no data to be used to record measurements in XP is stored.
- No other XP practice could be used to satisfy this specific practice.

Specific Practice 2.4 Communicate Results

Practice Result: Unsatisfied

- This specific practice is unsatisfied by the planning game practice as no data in XP is collected and analysed.
- If there were results to be discussed then they could be done after each iteration at the evaluation meetings with all stakeholders. This would allow action to be taken in the next iteration based on the results of the measurement data.

Project Monitoring and Control & The Planning Game

XP Practice – The Planning Game
CMMI-Dev Process Area – Project Monitoring & Control

Table 45: Project Monitoring and Control & The Planning Game Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of The Planning Game Practice
SG 1 Monitor Project Against Plan SP 1.1 Monitor Project Planning Parameters SP 1.2 Monitor Commitments SP 1.3 Monitor Project Risks SP 1.4 Monitor Data Management SP 1.5 Monitor Stakeholder Involvement SP 1.6 Conduct Progress Reviews SP 1.7 Conduct Milestone Reviews **SG 2 Manage Corrective Action to Closure** SP 2.1 Analyse Issues SP 2.2 Take Corrective Action SP 2.3 Manage Corrective Action	**(SG1)** Partially Satisfied **(SG2)** Partially Satisfied	Client presents desired features through use cases / storyboards. Developers produce cost estimates + requirement prioritisation. Initial schedule / plan produced before commencing iteration. Running software produced every 2-4 weeks. Customer involved in evaluating software. Problems / Changes Analysed & Discussed Feedback taken and incorporated into next 2-4 week iteration.

Specific Goal 1 Monitor Project Against Plan

Goal Result: Partially Satisfied

Specific Practice 1.1 Monitor Project Planning Parameters
Practice Result: Partially Satisfied

- This practice is partially satisfied as the planning game practice does record parameters such as project cost, schedule, and milestone progress, which can be discussed after each evaluation meeting.
- However the planning game practice does not monitor attributes of work products such as size, complexity, errors etc.

Specific Practice 1.2 Monitor Commitments
Practice Result: Satisfied

- The commitments of individuals can be monitored after each iteration in an evaluation stakeholder meeting where individuals work can be reviewed and checked for progress.

Specific Practice 1.3 Monitor Project Risks
Practice Result: Partially Satisfied

- Although XP and the planning game deal with risk by reviewing work after each iteration, no documentation is produced initially to record risks that may occur.
- By reviewing and producing software frequently, the planning game can deal with change and risk concurrently.

Specific Practice 1.4 Monitor Data Management
Practice Result: Unsatisfied

- The planning game practice does not set out plans to monitor any project data.

Specific Practice 1.5 Monitor Stakeholder Involvement
Practice Result: Partially Satisfied

- This specific practice is partially satisfied because the stakeholders are monitored during the meeting review at the end of the iteration. This is where their progress and contribution to the project can be analysed.
- To fully satisfy this specific practice, review documentation would need to be produced to record stakeholder issues and impacts.

Specific Practice 1.6 Conduct Progress Reviews
Practice Result: Satisfied

- This specific practice is satisfied by the planning game practice as reviews are performed at the evaluation stage and before the next iteration is commenced.

- These reviews can contain information on, evaluation, project status, project measurements, development problems etc.
- The aim of this specific practice is to allow the project to continue progressing and allow any issues that may arise to be communicated to the team so everyone is aware of project status. The planning game practice supports this by encouraging communication throughout the project.

Specific Practice 1.7 Conduct Milestone Reviews
Practice Result: Satisfied

- This specific practice is satisfied by the planning game practice as reviews before development and after evaluation in place by the planning game practice. They can be classed as milestones as they are important stages of the project and involve all stakeholders.
- The reviews are documented to record results, actions and decisions that are made at the milestones.

Specific Goal 2 Manage Corrective Action to Closure
Goal Result: Partially Satisfied

Specific Practice 2.1 Analyse Issues
Practice Result: Satisfied

- The planning game practice satisfies this specific practice as reviews are held during evaluation and before each new iteration. This is where issues can be analysed and discussed to find a solution.

Specific Practice 2.2 Take Corrective Action
Practice Result: Satisfied

- The planning game practice satisfies this specific practice as reviews are held during evaluation and before each new iteration. This is where issues can be analysed, discussed and a solution be developed to be put into place for the next iteration.

Specific Practice 2.3 Manage Corrective Action
Practice Result: Partially Satisfied

- The planning game practice satisfies this specific practice as reviews are held during evaluation and before each new iteration. This is where progress is discussed on problems to be solved.
- Documentation that would fully satisfy this specific practice would be used as a backlog to learn from mistakes.

Project Planning & The Planning Game

XP Practice – The Planning Game
CMMI-Dev Process Area – Project Planning

Table 46: Project Planning & The Planning Game Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of The Planning Game Practice
SG 1 Establish Estimates SP 1.1 Estimate the Scope of the Project SP 1.2 Establish Estimates of Work Product and Task Attributes SP 1.3 Define Project Lifecycle SP 1.4 Determine Estimates of Effort and Cost	**(SG1)** Partially Satisfied	Client presents desired features through use cases / storyboards. Developers produce cost estimates + requirement prioritisation.
SG 2 Develop a Project Plan SP 2.1 Establish the Budget and Schedule SP 2.2 Identify Project Risks SP 2.3 Plan for Data Management SP 2.4 Plan for Project Resources SP 2.5 Plan for Needed Knowledge and Skills SP 2.6 Plan Stakeholder Involvement SP 2.7 Establish the Project Plan	**(SG2)** Unsatisfied	Initial schedule / plan produced before commencing iteration. Running software produced every 2-4 weeks. Customer involved in evaluating software. Problems / Changes Analysed & Discussed
SG 3 Obtain Commitment to the Plan SP 3.1 Review Plans That Affect the Project SP 3.2 Reconcile Work and Resource Levels SP 3.3 Obtain Plan Commitment	**(SG3)** Unsatisfied	Feedback taken and incorporated into next 2-4 week iteration.

Specific Goal 1 Establish Estimates

Goal Result: Partially Satisfied

Specific Practice 1.1 Estimate the Scope of the Project
Practice Result: Partially Satisfied

- The planning game practice does not fully satisfy this specific practice as it performs little work in understanding the scope of the project. The only understanding comes form the client's requirements.

Specific Practice 1.2 Establish Estimates of Work Product and Task Attributes
Practice Result: Partially Satisfied

- The developers produce estimates of cost, effort, and schedule based on the client requirements thus the planning game partially satisfies this specific practice.
- To fully satisfy this specific practice, the technical approach should be determined, the methods to determine attributes of work products and estimates of tasks should be established, which the planning game does not include.

Specific Practice 1.3 Define Project Lifecycle
Practice Result: Unsatisfied

- This specific practice is unsatisfied as no lifecycle definition is chosen by the planning game practice.
- There are set activities and key points that are to be performed regardless of differentiation between projects.

Specific Practice 1.4 Determine Estimates of Effort and Cost
Practice Result: Partially Satisfied

- The developers involved in the planning game practice estimate effort and costs for the requirements that have been produced, which partially satisfies this specific practice.
- However to fully satisfy this specific practice, it requires an estimation rationale and totals for project cost and effort.

Specific Goal 2 Develop a Project Plan

Goal Result: Unsatisfied

Specific Practice 2.1 Establish the Budget and Schedule
Practice Result: Unsatisfied

- The planning game practice does not identify a budget for the project but does set out a schedule based on task estimates.

- Little time is spent on the project plan in XP as change is expected, thus more effort is spent developing than planning.

Specific Practice 2.2 Identify Project Risks
Practice Result: Unsatisfied

- Risks in the planning game practice are not identified prior to starting the project but are dealt with as they develop. The short development cycles allow testing and evaluations to occur that would highlight any risks to the project as they occur.
- Documentation produced: Risk Identification, Risk Impacts & Probability and Risk Priorities.

Specific Practice 2.3 Plan for Data Management
Practice Result: Unsatisfied

- Involves planning for managing data that is collected throughout the project to verify and validate why and how it should be collected. The planning game practice does not satisfy this specific practice.

Specific Practice 2.4 Plan for Project Resources
Practice Result: Unsatisfied

- The attributes of this specific practice are not taken into consideration by planning game practice. It assumes that all materials, tools, machinery, and staff are provided to complete the project.

Specific Practice 2.5 Plan for Needed Knowledge and Skills
Practice Result: Unsatisfied

- The attributes of this specific practice are not taken into consideration by planning game practice. No planning is performed into the amount of staff or the knowledge and skills required to complete the project.

Specific Practice 2.6 Plan Stakeholder Involvement
Practice Result: Unsatisfied

- The planning game practice involves many stakeholders but does not produce documentation such as rationale, roles and responsibilities, relationships and resources for the stakeholder etc.

Specific Practice 2.7 Establish the Project Plan
Practice Result: Unsatisfied

- As the planning game practice does not satisfy any of the above specific practices for this goal, it cannot be applied to this specific goal.

Specific Goal 3 Obtain Commitment to the Plan

Goal Result: Unsatisfied

Specific Practice 3.1 Review Plans That Affect the Project

Practice Result: Unsatisfied

- This specific practice is unsatisfied by the planning game practice as XP focuses on developing software frequently and fast where as this specific practice requires reviewing all plans and making sure they are consistent.

Specific Practice 3.2 Reconcile Work and Resource Levels

Practice Result: Unsatisfied

- After obtaining commitment from stakeholders, this specific practice requires the resources to be reviewed, to determine if any changes need to be made to budgets, schedules, requirements, agreements etc.
- The planning game practice focuses on development rather than documentation.

Specific Practice 3.3 Obtain Plan Commitment

Practice Result: Unsatisfied

- The planning game practice does not require to obtain a commitment from the work group / individual on whether they should complete the task.
- This is more of an organisational process for placing pressure on the individual / team to commit themselves to completing the task.

Software Quality Assurance

Process and Product Quality Assurance & Coding Standards

XP Practice – Coding Standards
CMMI-Dev Process Area – Process and Product Quality Assurance

Table 47: Process and Product Quality Assurance & Coding Standards Mapping

Specific Goal (SG) & Specific Practice (SP) Summary	Rating Criteria (for each goal)	Key Points Of Coding Standards Practice
SG 1 Objectively Evaluate Processes and Work Products SP 1.1 Objectively Evaluate Processes SP 1.2 Objectively Evaluate Work Products and Services	**(SG1)** Unsatisfied	Industry or Organisational Recognised Standard Written In One Style Code Looks Familiar From One Developer To Another Easier to Understand
SG 2 Provide Objective Insight SP 2.1 Communicate and Ensure Resolution of Noncompliance Issues SP 2.2 Establish Records	**(SG2)** Unsatisfied	

Specific Goal 1 Objectively Evaluate Processes and Work Products
Goal Result: Unsatisfied

Specific Practice 1.1 Objectively Evaluate Processes
Practice Result: Unsatisfied

- The coding standards practice does not satisfy this specific practice as it only deals with coding, where as this specific practice focuses on the processes that the organisation have in place to create work products.

Specific Practice 1.2 Objectively Evaluate Work Products and Services
Practice Result: Unsatisfied

- The coding standards practice does not satisfy this specific practice as it only deals with coding, where as this specific practice focuses on evaluating the designated work products and services against the applicable process descriptions, standards, and procedures.

Specific Goal 2 Provide Objective Insight
Goal Result: Unsatisfied

Specific Practice 2.1 Communicate and Ensure Resolution of Noncompliance Issues
Practice Result: Unsatisfied

- Evaluations are performed to discover noncompliance issues which reflect a lack of adherence to applicable standards, process descriptions, or procedures in this specific practice.
- The coding standards practice does not satisfy this specific practice because of the sole focus on coding.
- There is also no objective insight used, the coders are the ones setting the standard of the development team and no one else views the code other than developers.
- Pair programming practice could be used to improve the mapping by providing objective insight.

Specific Practice 2.2 Establish Records
Practice Result: Unsatisfied

- The coding standard practice does not satisfy this specific practice because no records are established, which could be used as lessons learnt.
- Work products produced to satisfy this specific practice include evaluation logs, quality assurance reports, status reports of corrective actions and reports of quality trends.