Requirements Engineering and Management Effects on Downstream Developer Performance in a Small Business

Findings from a Case Study in a CMMI/CMM Context

by
Chantelle Gellert

A thesis
presented to the University Of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2021

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Abstract— This thesis is a case study explaining how I tried to improve the requirements engineering process at company X (not its real name), a small software development company in Waterloo, ON, Canada. I assessed X's practices and standards using the Capability Maturity Model Integration (CMMI) and the Requirements Definition and Management (RDM) Maturity Model (RDMMM). I chose CMMI because it defines and measures a company to assess its maturity as an organization. Higher levels of CMMI have been found to have a correlation with the better success of projects, with regards to delivering the product on-time, on-budget, and on function. For analysis, initial measurements of the company's performance were gathered to compare results in order to measure X's process improvements. Six common performance metrics were analyzed: error density, software development productivity, percentage of rework, cycle time for the completion of a typical software project, schedule fidelity, and error detection effectiveness. In addition, I gave a questionnaire to X's employees based on Ellis's RDM which is a process for defining, documenting, and maintaining documents that take its reference point from empirical studies on the effectiveness of CMMI. [19] This case study's survey questions were used to elicit the data necessary to answer whether higher levels of the RDMMM in strategic projects lead to the success of projects at X. The different levels of RDMMM within the company were measured by comparing the questionnaire results taken in 2017 and 2019. Many of the conclusions and the results of this paper are based on the interviews and personal statements from employees at X about their experience in software development.

Acknowledgment

The completion of this case study was possible with help from participants at X.

Special thanks to my Dad, my Mom, my Professor, Daniel Berry, my friends, Team Winndelorean, my Auntie Pat, my Ryan, my Nova, and my family for their support.

Table of Contents

List of Figures

List of Tables

1 Introduction

In 2017, there were over 3,000 registered computer software companies in Ontario, according to Manta Statistics [26]. It is difficult for a small software company to get established. A substantial factor affecting a company's entering the software market is the quality of the company's software products. A high-quality product is directly dependent upon its development and maintenance process. Some software companies have realized that better processes result in better products. Companies have started to look for ways to improve the processes involved in producing software. Software processes are concerned with how the developers actually write code, and they involve complex interrelationships between a company's culture, technology, and economy. These processes are difficult to standardize because each company is unique. The application of software solutions has extended into various research fields and has increased the range of human accomplishments. Software solutions have replaced manual solutions, improved banking processes, created new forms of entertainment, served to enhance research, and more [1]. It remains difficult for a small company to enter the software market because of the high cost of developing software. Typically, a small software company's highest cost is developing software. A major part of that cost has been the part of the process called Requirements Engineering (RE) [7, 13,19,22]. So focusing on RE should have a major impact.

By 2017, RE was still an emerging field of study that had not been perfected, with many unknown possibilities yet to be discovered. RE is designed to cover all aspects of the software development process including: discovery, documentation, and software product development maintenance. RE techniques include: interviews, prototypes, studies, and standards, but these techniques are often poorly implemented by companies, which results in unfavorable outcomes [12]. Even with an established process for representing a system's components, and interactions, problems are often discovered only at very late stages of the life cycle. The later a problem is discovered in the development life cycle, typically the higher the relative cost to fix the problem for the simple reason that changing code, which is highly connected to itself and other artifacts, is a lot more complex than changing a few natural language sentences in a requirements document.

A software development company should invest in improving its processes to produce better software products. Capability Maturity Model Integration (CMMI) is a model that helps organizations streamline process improvement. CMMI consists of two types of CMMI levels, capability, and maturity. This case study does not discuss the capability levels it discusses only the maturity levels.

Requirements Definition and Management (RDM) is what happens during RE[19]. RDM is the elicitation, analysis, specification and validation of requirements, and constraints. The RDM Maturity Model (RDMMM) is a maturity model specifically designed to help an organization develop better RDM. This case study's RDM questionnaire is found in Section 9.1. A RDMMM level is derived from the median of a specified set of participants' numerical values obtained from the RDM questionnaire. RDMMM level is the assessed strength of a company's capabilities supporting RDM on a Likert scale from 1 to 5. RDMMM Level 1, meaning "very low ability to accomplish good RDM", RDMMM Level 3, meaning "neutral ability to accomplish good RDM", and RDMMM Level 5, meaning "very high ability to accomplish good RDM".

A CMMI level and an RDMMM level are well-defined evolutionary plateaus towards obtaining a mature software process. Each maturity level for each of CMMI and RDM provides milestones towards building a foundation for continuous process improvement. I chose CMMI as a guideline to monitor development performance because CMMI's structure is well defined, CMMI is known in the SE community and CMMI focuses on integrating multiple disciplines to help organizations streamline SE process improvement. I chose RDMMM because it focuses on the RE process and measures its success through projects which is aligned with how I want to measure developers performance on projects.

X is a specific branch of an American company that is located in Waterloo, Ontario, Canada. It works in nearly complete isolation from its parent company and sister branches. I investigated X's improvement efforts between June 2017 and October 2019. This case study reports the benefits of X's improvement efforts. I aimed to observe the effects of X's improvement efforts on downstream developer performance by tracking its CMMI level and RDMMM level.

I assessed the effects of X's improvement efforts on downstream developer performance by using the following steps:

1. Conduct an RDMMM level questionnaire.
2. Conduct a performance metrics questionnaire.
3. Observe X's operations and conduct interviews.
4. Assign a CMMI level.
5. Repeat steps 1-4 approximately two years later.
6. Compare the two results and conclude.

The performance metrics used during Step 2 are: error density, productivity, rework time, cycle time, schedule fidelity, and error detection effectiveness [28].

I conducted this case study to see if X's overall improvement efforts would result in a higher CMMI level, a higher RDMMM level, and higher developer performance metrics. This case study reveals findings from monitoring a company that I assessed to have higher CMMI levels, higher RDMMM levels, and higher developer performance metrics over the course of the two years observed. While a correlation can be observed it's not clear what caused what.

The results of this case study are essential for designing more studies of how to accrue the actual benefits of higher CMMI levels and higher RDMMM levels.

2 Background

2.1 Company Sizes

The definition of a small and large company can depend on the country in which the company is located. The United States Small Business Association considers a company with fewer than 1500 employees a small business [38], while Industry Canada considers a company with fewer than 100 employees a small business [39]. The size of a company affects how the company responds when an RE issue arises.

A large company generally has a single well-defined process adopted across the entire company, making it difficult to adapt to changing needs. If a company has a vague or poorly defined process to approaching requirements, various solutions to the same problem could be developed, leading to confusion, redundancy, or errors.

A company can experience different office cultures. In different office cultures, people may have different assumptions, values, and beliefs. For example, an office based in China could have an office culture different from one in Canada. Even within one location, a company may experience different office cultures for different roles. At X, the cultures of Quality Assurance (QA) workers and developers were vastly different in 2017, and I observed the two groups segregated into their respective social groups even when working on the same team.

A large software company generally has personnel dedicated to improving developer performance. A small software company typically hires more software developers, instead of personnel dedicated to improving developer performance. X in 2017 did not have personnel dedicated to improving developer performance.

Many projects within the same company can have completely different domains. For example, Microsoft created a software product called Microsoft Word, used as a document editor. Microsoft also created a gaming console known as the Xbox [14]. A small company typically focuses on building products in only one domain.

A good understanding of customer requirements and development technologies could prevent catastrophic consequences for the company. For example, BlackBerry developed a trusted security system with its mobile phones that corporations' Information Technology (IT) departments could control; however, BlackBerry's software for the mobile phone was harder to add on to for 3rd-party developers. Third-party developers could not manipulate BlackBerry's software to fit their corporate needs. The inability to alter BlackBerry's mobile phone software to meet third-party corporate needs led to a two-device problem when employees would bring two phones to work, one for personal use and one for company use. Google created software solutions to fix the two-device problem, but Google's solutions were not as secure as BlackBerry's software. Although Google was smaller in the mobile phone market than Blackberry, Google came up with its solutions for the two-device problem. Google's understanding of customer requirements and developmental technologies allowed it to become a strong competitor in the mobile phone market [11].

It is essential to understand when to create a product in-house, and when to purchase a third-party product. A smaller company is more likely to purchase products created by other companies than creating products in-house because of the high production costs associated with creating products. For example, a software developer uses an Integrated Development Environment (IDE) to write code. [40] Microsoft offers free IDEs to download and use. It would be more cost-effective for a company to use Microsoft's free IDEs instead of building its own. Even though much software is free and open-source a company will still have to consider the development cost of creating in-house products versus licensing or purchasing third-party products. The company will have to also consider if the free software meets its requirements before it adapts the free software.

A software development company of any size could take part in a concept of the technological world called "dogfooding" [15]. Dogfooding is when a company uses the software it creates. It is essential to use a product while developing it to understand its practical application. Using dogfooding for any size company is beneficial, as it encourages all employees to use and report issues with a product, not just its QA members.

A large company will often have more employees responsible for a knowledge domain than a small company. A small company is at a higher risk of losing essential resources for a knowledge domain than a large company when employee transitions or departures occur.

Lastly, RE processes must be defined based on the given situation and individual company [10]. When a company wants to integrate better RE methods, it is essential that the RE process be aligned with its business context, objective, and strategy. The company's objective should be the driving force behind process improvement efforts, as it provides the necessary background to guide the process improvement activities. RE process improvement should be undertaken to help a company achieve its objectives. For example, a company's objective could be to increase its total income by 10% over the next two years.

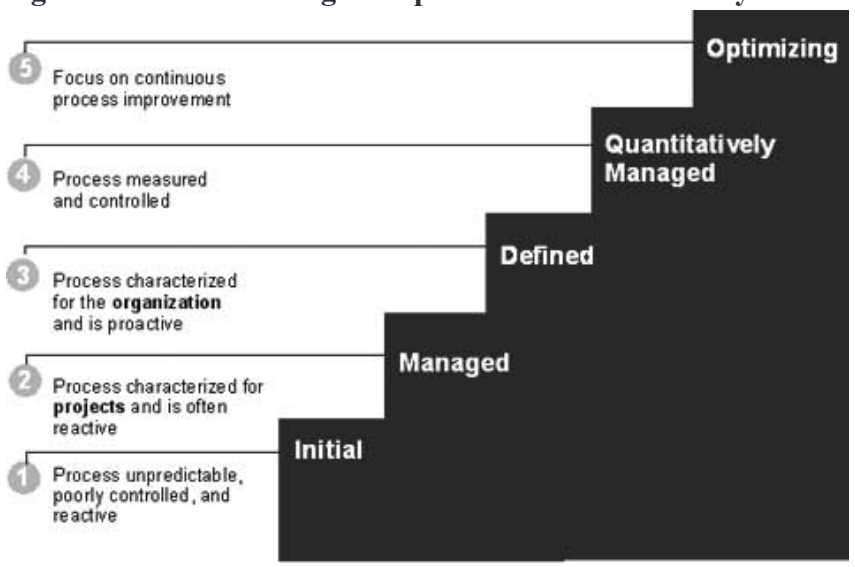
For the duration of this case study, X did not have enough interaction with its parent company and sister branches to consider it more than a small company. In 2017, X had approximately 70 employees, which meets the small business employee size criterion for both Canada and the United States. X grew from approximately 70 employees in 2017 to almost 100 employees in 2019, which still meets the small business employee size criterion for both Canada and the United States.

2.2 Capability Maturity Model Integration (CMMI)

Capability Maturity Model Integration (CMMI) is a well-defined guideline for process integration and product improvement used worldwide as a benchmark for software quality. CMMI is a company assessment model that raises awareness of areas for improvement. These areas are identified to improve productivity, performance, cost, and stakeholder satisfaction. CMMI focuses on long-term and consistent solutions that do not rely on employee heroics for last-minute miracles. CMMI focuses on total-system problems. CMMI is geared towards enterprise-wide process improvement instead of single-discipline models [16].

CMMI levels are well-defined evolutionary plateaus. Each CMMI level provides milestones towards building a foundation for continuous process improvement. CMMI guidelines advise that CMMI levels should not be skipped because each maturity level provides some necessary foundation for effectively advancing to the next CMMI level [31].

Figure 2.2.1 CMMI Staged Representation of Maturity Levels [31]



CMMI has five levels, as illustrated in Figure 2.2.1:

1. Initial
2. Managed
3. Defined
4. Quantitatively Managed
5. Optimizing

The following are descriptions of only the CMMI levels that this case study explores.

CMMI Level 1 - Initial

CMMI Level 1 is the baseline for a company to start. A company that is ad hoc and chaotic is considered to be at CMMI Level 1. The key indicators that a company is at CMMI Level 1 are:

1. The company relies on the competence and heroics of its employees rather than a proven process for success.
2. The company produces products that work; however, they often exceed budgets and miss project deadlines.

3. The company often overcommits, abandons processes during a crisis, and cannot reproduce its past success.

CMMI Level 2 - Managed
A company with a planned, performed, measured, and controlled process at the project level is considered to be at CMMI Level 2. The key indicators that a company is at CMMI Level 2 are:

1. The company's projects are managed by a process that is documented.
2. The company utilizes the process in place during times of stress.
3. The company's project's status is visible to management at defined points in the project's development life cycle.
4. The company's commitments are established with relevant customers and are revised as needed in a change-control process.

CMMI Level 3 - Defined
A company is considered to be at CMMI Level 3 when it has processes throughout its organization, and its processes are proactively managed. The key indicators that a company is at CMMI Level 3 are:

1. The company has achieved all key indicators at CMMI Level 2.
2. The company's processes are understood throughout the organization, encompassing standards, procedures, tools, and methods.
3. The company's processes performed across the organization are consistent. Minor differences are allowed.
4. The company's processes are typically described in great detail and are managed proactively.

X was not using the CMMI framework. However, I chose CMMI as a guideline to monitor development performance at X because CMMI's structure is well defined, and CMMI is known to be reputable in the RE community.

Typically, a small company would avoid implementing CMMI because of personnel limitations, the cost associated with getting started, overall maintenance, and the delay in recovering invested capital. Adopting the CMMI framework requires a substantial commitment of time and resources from a company. There are two primary choices in the CMMI realm to determine a company's status regarding process areas that need attention:

1. A formal standardized approach such as the Standard CMMI Appraisal Method for Process Improvement (SCAMPI).
2. A self-assessment such as CMMI Acquisition Model (CMMI-AM).

This case study uses CMMI-AM to evaluate X's process improvement changes. I chose CMMI-AM for this case study because it is cheaper to implement than SCAMPI.

2.3 Standard CMMI Appraisal Method for Process Improvement (SCAMPI)

When a decision has been made to follow CMMI, benchmarks to meet higher CMMI levels must be assessed. SCAMPI provides benchmark ratings with which to assess a company's CMMI level [9]. SCAMPI's benchmark ratings provide a company with a report card on its process improvement efforts, and to check its compliance with CMMI implementation. A CMMI assessment is meant to identify a company's strengths and weaknesses, and consolidate the company's process improvements on a sustainable basis. SCAMPI provides a company with guidelines to prioritize its CMMI improvement plans, focusing on improvements that are the most beneficial to its current CMMI level, and process capabilities. SCAMPI will derive a company's CMMI level based on its organizational unit, organizational scope, reference model scope, appraisal method type, and an appraisal of all team leaders' compatibility with their teams. SCAMPI guidelines recommend that a company reach CMMI Level 2 before conducting SCAMPI [5].

Since SCAMPI is used to find a company's readiness to adapt to CMMI, X's best course of action in 2017 was most likely to use SCAMPI to identify where to begin software development improvements; however, X did not conduct SCAMPI because of SCAMPI's high cost. Nevertheless SCAMPI's model of tracking benchmark performance metrics aligned with how I wanted to track performance improvement efforts.

2.4 Capability Maturity Model Integration Acquisition Model (CMMI-AM)

For a company in which process maturity is a new concept, a self-assessment offers an easy way to begin process improvement with minimal cost and effort. No minimum CMMI level achievement is needed before implementation. CMMI-AM helps a company find gaps in its current practices before exposing itself to the external scrutiny of a SCAMPI evaluation. CMMI-AM results can educate a company about acquisition modules and formal appraisal method requirements.

A self-assessment is conducted with a questionnaire given to employees anonymously. The responses are then aggregated, the mode taken and presented to the team for further discussion. CMMI-AM questions are purposefully devoid of process model terminology. Instead, CMMI-AM questions use language that should be more familiar and accessible for all employees to participate. Each CMMI-AM question uses a Likert scale from 1 to 10; 1 represents the statement on the left, and 10 represents the statement on the right. CMMI-AM's goal is to get the most diverse representative response possible to avoid skewing the results. Therefore, project managers, senior engineers, junior engineers, user experience engineers, team leaders, contractors, and more are encouraged to participate.

The CMMI-AM creators noted a tendency for employees to rate their company higher than an external appraisal team, which explains why CMMI-AM recommends the mode responses for each process area to be near 10 before paying for a SCAMPI appraisal [29].

A considerable concern with the self-assessment process is that evidence is not required to prove the accuracy of a respondent. Interviews and focus sessions are highly recommended in order to determine specific examples of why ratings were chosen. Self-assessment results can

depend on the respondent's state of mind the day of the self-assessment. Thus, having many respondents and regularly conducting self-assessments is recommended.

I conducted two self-assessments at X:

1. Performed by multiple employees in 2017 and 2019 for evaluating X's RDMMM level
2. Performed by myself in 2019 for evaluating X's CMMI level and SCAMPI readiness.

The first of these self-assessments is found in Section 9.1. The second of these self-assessments is found in "Self Assessment and the CMMI-AM - A Guide for Government Program Managers" [29].

2.5 Capability Maturity Model (CMM) and Key Process Area (KPA)

CMMI evolved from the Capability Maturity Model (CMM). CMM focuses on incorporating components of individual disciplines, whereas CMMI focuses on integrating multiple disciplines when needed [16]. A company can use CMM as a benchmark to measure its software process maturity using CMM Key Process Areas (KPAs) [31]. This case study uses CMM KPAs to evaluate X's process improvement changes.

KPAs need to be fulfilled in order to move up CMM levels. CMM levels are not the same as CMMI levels. Transitioning from one level to another is a CMM Level Transition (CMMLT) [28].

Table 2.5.1: Major Characteristics of CMM level [28]

CMM Level	Type	Major Characteristics
1	Initial	The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
2	Repeatable	Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3	Defined	The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the company. Projects use an approved, tailored version of the company's standard software process(es) for developing and maintaining software.
4	Managed	Detailed measures of the software process and product quality are collected. Both the software process and the products are quantitatively understood and controlled.
5	Optimizing	Continuous process improvement is facilitated by quantitative feedback from the process and from piloting innovative ideas and technologies.

Table 2.5.2: KPAs for Respective CMM level [28]

CMM Level	Focus	KPAs
1	Competent people and heroics	
2	Project management processes	Requirements management Software project planning Software project tracking and oversight Software subcontract management Software quality assurance Software configuration management
3	Engineering processes and Organizational support	Organization process focus Organization process definition Training program Integrated software management Software product engineering Intergroup coordination Peer reviews
4	Product and process quality	Quantitative process management Software quality management
5	Continuous process improvement	Defect prevention Technology change management Process change management

CMM levels start at Level 1, where no KPAs are required. For *m* from 1 to 5, a company transition from CMM Level *m* to CMM Level *m*+1 by succeeding in and maintaining all KPAs required for CMM Level *m*+1 while maintaining all KPAs, if any, required for CMM Levels 1 through *m*.

This case study focuses on the CMM Level 2 and CMM Level 3 KPAs defined in Table 2.5.1 and Table 2.5.2, respectively.

To achieve a CMM level, each of the KPAs for that level must be satisfied. Each KPA consists of goals. A KPA goal signifies the scope, boundaries, and intent of the KPA. A KPA goal summarizes the key practices of the KPA, and can be used to determine whether an organization or project has effectively implemented the KPA. To satisfy a KPA, each of the goals for the KPA must be satisfied.

2.5.1 CMM Level 2 KPAs

Each of the following headers refers to a single KPA found in Table 2.5.2 for CMM Level 2 [31].

The following defines each CMM Level 2 KPA and its goals. If I felt that a goal's definition is ambiguous, I added clarity below the goal to elaborate on what X needed to establish to satisfy the goal. All goals mentioned below come from "Capability Maturity Model for Software, Version 1.1" [31].

Requirements Management
Requirements Management is to manage RE by establishing an understanding between a customer and a software project in order for the software project to fulfill the customer's requirements. An agreement made with a customer is a basis for planning (as described below in "Software Project Planning"), and managing (as described below in "Software Project Tracking and Oversight") of a software project. The control of the agreement with the customer depends on following an effective change control process (as described below in "Software Configuration Management").

Goal 1 System requirements allocated to software are controlled to establish a baseline for software engineering and management use.

Goal 2 Software plans, products, and activities are kept consistent with system requirements allocated to software.

Software Project Planning
Software Project Planning is to institute plans for performing software engineering and for managing a software project. These plans are the foundation for managing the software project (as described below in "Software Project Tracking and Oversight"). The plans are fundamental for effective project management to be implemented.

Goal 1 Software estimates are documented for use in planning and tracking the software project.

Goal 2 Software project activities and commitments are planned and documented.

Goal 3 Affected groups and individuals agree to their commitments related to the software project.

Software Project Tracking and Oversight
Software Project Tracking and Oversight is to institute reasonable visibility into the progress of a software project in order for management to take practical actions, when the software project deviates substantially from its current project's plans.

Goal 1 Actual results and performances are tracked against the software plans.
Clarity: *I will be looking to see if X has visual proof of a system for tracking.*

Goal 2 Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans.

Goal 3 Changes to software commitments are agreed to by the affected groups and individuals.

Software Quality Assurance
Software Quality Assurance is to allow management visibility into a process used by software products and projects. A company's software quality assurance member reviews all products for managing requirements and reports results. This member also takes part in requirement change negotiations.

Goal 1 Software quality assurance activities are planned.

Goal 2 Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively.

Goal 3 Affected groups and individuals are informed of software quality assurance activities, and results.

Goal 4 Noncompliance issues that cannot be resolved within the software project are addressed by senior management.

Software Configuration Management
Software Configuration Management is to institute and maintain a software project's product integrity throughout the project's life cycle. Project configuration management focuses on how to set up and configure a project.

Goal 1 Software configuration management activities are planned.

Goal 2 Selected software work products are identified, controlled, and available.
Clarity: Selected software work products are the products developed at the company.

Goal 3 Changes to identified software work products are controlled.

Goal 4 Affected groups and individuals are informed of the status and content of software baselines.

Software Subcontract Management
Software Subcontract Management is to choose qualified software subcontractors and manage them effectively. All concerns for the previously mentioned KPAs apply to subcontractors when applicable.

Goal 1 The prime contractor selects qualified software subcontractors.

Goal 2 The prime contractor and the software subcontractor agree to their commitments to each other.
Clarity: *I will look for clearly written commitments signed by both parties of each other's expectations and timelines.*

Goal 3 The prime contractor and the software subcontractor maintain ongoing communications.

Goal 4 The prime contractor tracks the software subcontractor's actual results and performance against its commitments.

2.5.2 CMM Level 3 KPAs

Each of the following headers refers to a single KPA found in Table 2.5.2 for CMM Level 3 [31].

The following defines each CMM Level 3 KPA and its goals.

Organization Process Focus
Organization Process Focus is to establish a company's responsibility for software process activities that focus on improving its overall software process capability. This process primarily results in creating (as described below in "Organization Process Definition") and applying (as described below in "Integrated Software Management") software process assets.

Goal 1 Software process development and improvement activities are coordinated across the organization.

Goal 2 The strengths and weaknesses of the software processes used are identified relative to a process standard.
Clarity: I will not consider this goal accomplished until the Organization Process Definition goals below are met

Goal 3 Organization-level process development and improvement activities are planned.

Organization Process Definition
Organization Process Definition is to develop and maintain a usable set of software process assets that improve process performance across projects, and provide a basis for cumulative, long-term benefits to a company. These assets build a stable foundation utilized in mechanisms such as training (as described below in "Training Program").

Goal 1 A standard software process for the organization is developed and maintained.

Goal 2 Information related to the use of the organization's standard software process by the software projects is collected, reviewed, and made available.

Training Program
Training Program is to develop an individual's skills, and increase their knowledge enabling them to perform their roles more effectively, and efficiently. Training is a company-level responsibility, but software projects should also identify which training is needed.

Goal 1 Training activities are planned.
Clarity: A training schedule will be provided.

Goal 2 Training for developing the skills and knowledge needed to perform software management and technical roles is provided.

Goal 3 The software engineering tasks are defined, integrated, and consistently performed to produce software.

Integrated Software Management
Integrated Software Management is to unite software engineering, and management activities into defined software processes and assets (as described above in "Organization Process Definition"). This union of activities recognizes a project's business environment and technical needs (as described below in "Software Product Engineering"). Integrated Software Management is an extension of Software Project Planning and Software Project Tracking, and Oversight from CMM Level 2 KPAs.

Goal 1 The project's defined software process is a tailored version of the organization's standard software process.
Clarity: I will not consider this goal accomplished until the Organization Process Definition goals above are met.

Goal 2 The project is planned and managed according to the project's defined software process.

Software Product Engineering
Software Product Engineering is to perform a well-defined engineering process consistently. Software Product Engineering integrates all software engineering activities to produce software products effectively and efficiently. Software Product Engineering describes a project's technical activities, such as its code, requirements, designs, and tests.

Goal 1 The software engineering tasks are defined, integrated, and consistently performed to produce the software.

Goal 2 Software work products are kept consistent with each other.

Intergroup Coordination
Intergroup Coordination is to establish a process for a software engineering team to participate collectively such that a project can better satisfy customer needs effectively and efficiently. Intergroup Coordination is the interdisciplinary aspect of Integrated Software Management that extends beyond software engineering. A software engineering team's interactions with other members must be coordinated and controlled through an integrated process.

Goal 1 The customer's requirements are agreed to by all affected groups.

Goal 2 The commitments are agreed to by the engineering groups and the affected groups.

Goal 3 The engineering groups identify, track, and resolve intergroup issues.

Peer Review
Peer Review is to establish a process to remove defects from a software work product early and efficiently. Peer Review allows reviewers time to better understand their software products. Peer Review can be implemented in various ways, such as structured walkthroughs, Fagan-style inspections, or various other collegial review methods.

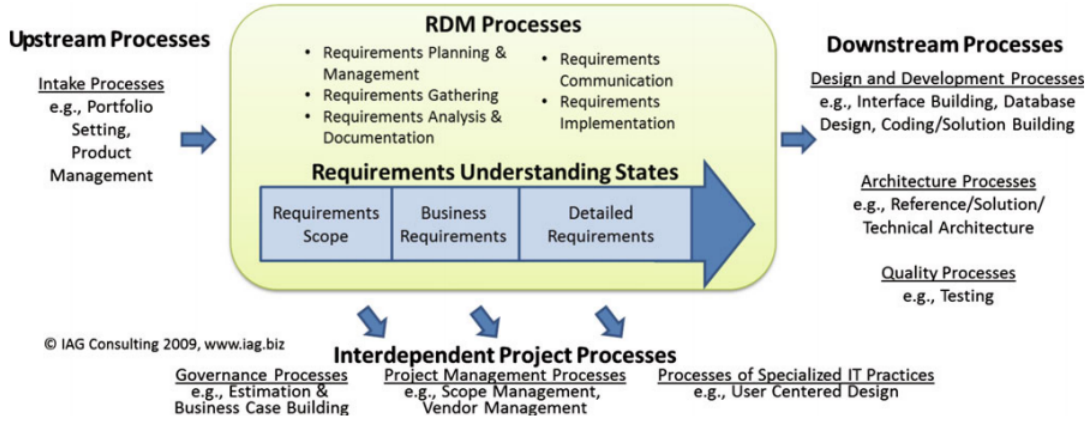
Goal 1 Peer review activities are planned.

Goal 2 Defects in the software work products are identified and removed.

2.6 Requirements Definition and Management Maturity Model (RDMMM)

Requirements are not a document. Requirements are a process that produces and maintains documents referred to as Requirements Specifications (RSs). This process is what this case study calls "Requirements Definition and Management (RDM) process", as illustrated in Figure 2.6.1 from IAG Consulting [19].

Figure 2.6.1: IAS Consulting's RDM framework



RDM takes its reference point from empirical studies on CMMI's effectiveness [19]. RDM is specifically designed to develop better RE. The maturity of a company's RDM process is evaluated through the RDMMM.

To summarize the key elements of the RDMMM.

1. Requirements are a process—the RDM process.
2. A high-quality RDM process must be defined and supported by capabilities that allow the process to be implemented.
3. An RDM process must be implemented and mature to have value.

Quantifying the impact of a company's requirement documents on its development processes relies on establishing the RDMMM measures and determining how the company performs at the RDMMM levels. By assessing the strength of the company's capabilities supporting RDM, its RDMMM level can be determined. The strength of the company's capabilities supporting RDM is considered in 5 levels: RDMMM Level 1, meaning "very low ability to accomplish good RDM", RDMMM Level 3, meaning "neutral ability to accomplish good RDM", and RDMMM Level 5, meaning "very high ability to accomplish good RDM". Evidence dictates that higher RDMMM levels predict a company's strategic commercial application (CA) development projects' success [19].

2.7 Jira

Atlassian created many software products designed to help a company with its RE process, including the Jira issue tracker. An issue tracker provides templates, guidelines, and documentation on how each phase of the RE processes should be executed using an issue tracker. Having a software-based solution for the RE process allows all documentation to be presented to all stakeholders simultaneously sharing access. Jira represents an RS as what is commonly referred to as a Jira task [32].

Jira supports functionality for Jira tasks to be placed on a Jira board. A Jira board is a visualization of what tasks are currently being worked on for a given period. When a Jira board is active in the present, it is referred to as a Sprint. When a Jira board is not active but will be active in the future, it is referred to as a backlog [32].

Jira allows for (1) contact among stakeholders across physical distances, (2) flexible communication times, and (3) low-latency communication. Jira does not provide private one-to-one communication. Jira serves as a group message board, allowing individuals to communicate by updating Jira tasks and posting comments. Comments are shared among all individuals with the required security access for a given Jira task. For many years, technology has been creating better strategies to replace person-to-person interactions; however, there are still shortcomings to pure text-based communication. Jira allows users to add various media types such as picture and video files to help with visualization.

A sizable shortfall with in-person communication is that information can easily be forgotten. Jira stores history so that information is not easily lost or forgotten. Jira allows stakeholders to view what changes were made and when. Jira's popular features include: roadmaps for illustrating the overall big picture, Jira boards for updating requirements as they change, a workflow engine for designing the system at a high level, and development and operations (DevOps) metrics integrated for tracking progress.

Before 2016, X used two different issue trackers called Fogbugz and Target Process. Because of X's poor experience with Fogbugz's and Target Process's poor user interface, X switched to Jira in 2016.

3 Research Method

3.1 The Company and Challenges in Requirements Practice

I conducted a case study that examined X, a United States company's satellite office in Waterloo, Ontario, from June 2017 through October 2019. X was a research and development branch, and worked independently of its United States parent company. (Only the Human Resource component was shared between them.) Agreements between X's parent company's business unit and X were made for any new software features X was to release. Agreements were to consider product strategy requirements regarding technology and engineering direction in line with the company's strategic plan, and market needs. In addition, agreements were to consider current and potential customer needs. In 2017, X was not completing new software features because the agreement language was ambiguous.

X did not have any formal experience regarding any maturity model in 2017. X's CMMI situation during 2017 and 2019 was dire and is summarized in Section 4.3.

For more context, X in 2017 had the following substantial issues:

- One-line statements were passed off to X's developers as requirements, that often required extensive knowledge of X's systems for context.
- There was no direct communication line for X's developers to end-users to clarify requirements, nor was it clear who was responsible for signing off on requirement completion.
- There was poor clarity on X's mock-ups when given to X's developers. Each mock-up given to the developer was presented as a static picture with no footnotes, and the developer relied heavily on his or her own interpretation of the mock-up's functionality.
- There was little traceability with regard to requirements. Many records were missing details such as its priority, rationale, and responsibility of a requirement request.
- X's software development projects often experienced requirements creep, leading to frustration within the development team.

I offered to help and X accepted. There was no formal contract for this case studies work with X but I was given the ability to observe and participate in the day to day operations at X and to present recommendations and findings to the director of research and development in written format. There was no upfront commitment by X to implement my recommendations.

X has a ranking system for all employees represented by each employee's job title. An employee's job title indicates an employee's value and skills that they bring to X. The job titles for developers at X, in order from lowest to highest ranking, are: "Co-op", "Associate Developer", "Staff Developer", "Senior Developer", and "Architect".

A salesperson is an employee at X's parent company who sells software that X develops. X does not participate in the sales of its products.

A manager's responsibilities can include managing developer priorities, managing hardware resources, and coordinating customer requirement changes.

A developer's primary responsibility is to develop code.

3.2 Case Study Motivation

This case study reports on benefits from X's improvement efforts. I aimed to observe any effects from X's improvement efforts on downstream developer performance, X's CMMI level, and X's RDMMM level. I assessed the effects of X's improvement efforts by using the following steps in August 2017 and again in July 2019:

1. Observe operations, conduct informal interviews and draw conclusions about X's CMMI level and RDMMM level.
2. Gather performance metrics on selected projects.

The first step used the RDM Questionnaire described in Section 9.1. The second step used the Performance Metric Data Table described in Section 9.2.

The RDM Questionnaire solicited input about each participant's perception of X's RDMMM level and the participant. The Performance Metric Data Table solicited performance metrics for a project. The performance metrics solicited were error density, productivity, rework time, cycle time, schedule fidelity, error detection effectiveness, and project success [28].

Having a minimum of a two-year gap before conducting the above steps again was essential for measuring long-term benefits for the following reasons:

- Appraisal results are typically valid for up to three years [33].
- Strategic planning typically looks two to five years into the future [31].
- On average, it takes about two years for a CMMI level transition [27].
- X typically takes at least a year before a substantial production release occurs.

I collected data by conducting a questionnaire, inspecting requirement documents, observing X's quality assurance process, observing X's requirement analysis process, observing negotiation sessions, and gathering performance metrics on X's projects. I also conducted informal interviews with X's developers, QA members, co-op students, and managers.

3.3 Design of this Case Study

All projects, Jira tasks and participant names from X are fictional. No actual project or participant names are used in this report.

3.3.1 Data Collection Procedures

I used three main methods for data collection. First to evaluate the RDM process at X, I gave a questionnaire to volunteers at X to learn their subjective opinions about X. Second, to evaluate developer performance I filled out the performance metrics data table. Metrics gathered from X's Jira database included: Error density, rework time, cycle time, schedule fidelity, and error detection effectiveness which is subjectively taken but objectively created by X's employees. Third, to evaluate X's CMMI level, I used my subjective opinion based on experiences, interviews, and other data gathered.

3.3.1.1 Questionnaire Respondents

In 2017, the RDM Questionnaire was sent to over thirty people, of which ten responded, accounting for roughly 15% of X's work population. In 2017, there was no participation in the RDM Questionnaire by X's upper management, such as IT executives, managers, and professionals concerned with developing CAs. In 2017, the highest-ranking participant in the

RDM Questionnaire was a senior developer. In 2017, co-op students made up 50% of the RDM Questionnaire's participants.

In 2019, I sent again the RDM Questionnaire to over thirty people, of which twenty responded, accounting for roughly 20% of X's work population. In 2019, there was no participation in the RDM Questionnaire by X's upper management. In 2019, the highest-ranking participant in the RDM Questionnaire was an architect. In 2019, co-op students made up 40% of the RDM Questionnaire's participants.

Three participants completed the RDM Questionnaire twice, once in 2017 and once in 2019.

I filled out the Performance Metric Data Table for the Project Octopus twice, once in 2017 and once in 2019. I filled out the Performance Metric Data Table for the projects Moose, Crocodile, and Jellyfish once in 2019. In 2017, Project Octopus's lead, Participant M, worked primarily on Project Octopus. In 2019, Participant M left Project Octopus and shifted his focus to projects Moose, Crocodile, and Jellyfish. In 2019, I fille d the Performance Metric Data Table for Project Octopus to compare to its Performance Metric Data Table performed in 2017. In 2019, I filled the Performance Metric Data Table for projects Moose, Crocodile, and Jellyfish to observe project performance changes based on varying oversight from Participant M.

I conducted informal interviews with many different employees at X between 2017 and 2019 to understand X's improvement efforts. IT executives, managers, and engineers concerned with developing CAs participated in the interviews. The interviews were open-ended but provided an opportunity to explore how the RE process changed over the research period.

Interviews were candid and confidential. Participants were encouraged to be honest about their opinion of X's RDM. As there was no influence from upper management beyond allowing this case study to be conducted, participants' opinions were not to be publicized, and participants were to remain anonymous, I regarded the participants' opinions as honest opinions of X's RDM. I was aware of all associations between interviewed participants' identities and their projects. To further guarantee anonymous involvement, the RDM Questionnaire was conducted on a third-party Website on which a participant could confidently provide information in a private environment.

3.3.1.2 Challenges / Threats to Validity

The following are some challenges and threats to this case study's validity:

- Without financial compensation as an incentive or direct company involvement, finding participants was a challenge.
- It was challenging for me to understand X's projects from end to end entirely with little upper management involvement.
- A high percentage of the RDM Questionnaire participants were co-op students. I believe lower-ranking employees such as co-op students would have a less accurate perception of X's RDMMM level than higher-ranking employees. I believe a high percentage of low-ranking participants in the RDM Questionnaire could skew this case study's conclusions on X's RDMMM level farther away from X's true RDMMM level.

- Due to co-op students' short-term employment and full-time employees moving on to various other companies, I could not keep the same participants each time the RDM Questionnaire was conducted.
- The RDM Questionnaire sample size varied between 2017 and 2019. In 2017, the sample size was ten, and in 2019, the sample size increased to twenty because X's employees had greater interest and familiarity with this case study.
- Due to a high turnover rate, it was challenging to find participants familiar with X's products' history, and most importantly, with X's RE process.
- Each of the RDM Questionnaire participants was an engineer working on research and development. The RDM Questionnaire did not have participants from the Sales and Human Resources and similar departments.

3.3.2 Aspects Investigated

This section describes aspects of X that were investigated. Initial data gathering was conducted through a questionnaire (see Section 9.1), filling out a performance metric data table (see Section 9.2), and interviews.

3.3.2.1 The RDM Questionnaire

The RDMMM level portion of the RDM Questionnaire is a reproduction of questions included in the first study done in Ellis's paper, titled "Quantifying the Impact of RDM Process Maturity on Project Outcome in Large Business Application Development" [19]. I used the RDM Questionnaire to quantify any impacts an RDMMM level change has on a project's success.

Even though X is considered a small company, it can still suffer from similar RDM process faults found in a large company. The RDM Questionnaire was used to elicit the data necessary to answer this case study's question of whether a higher RDMMM level for developing strategic projects would lead to higher project success at X. I investigated X's business requirements documentation quality, X's RE process, and day-to-day operations at X to comprehend the RDM Questionnaire results.

A strategic project satisfies three main restrictions [19]:

1. The project's budget must be more than $250,000, including development required hardware, and external services.
2. The project must involve software development or application implementation.
3. The project must deliver business capability or software functionality substantially different from what existed before the project.

The three restrictions above aim to exclude projects that do not have a moderate amount of complexity, that do not develop infrastructure or roll out new technology, and projects that are in maintenance, bug-fixing, or platform migration stages. I chose to observe projects whose conduct was substantially different from how X typically conducted business in 2017.

In 2017, X focused on developing new technologies over several years, which resulted in X devising many strategic projects. This case study's participants worked on strategic projects. Co-op students and full-time employees worked on the same projects, allowing for a more reliable analysis than if they had been working on different projects. A task for a strategic project at X typically stretched over several days. I adjusted the RDM Questionnaire focus

from strategic projects to strategic project tasks because a typical co-op student's contract is four months long, and strategic projects typically stretch over several years.

A story point is a subjective unit of measure a project team decides for expressing an estimate of the overall effort required to fully implement a task. A story point at 1 typically was a task that would require little time to complete typically only needed a few lines of code changed. A story point at 3 typically would require work in more than one project file. A story point at 5 would typically involve senior developers, coordination with multiple teams and research before the task could begin. Therefore, a task that has been awarded at least five story points is considered a strategic task, in keeping with the usual definition given above.

Goal Definition: The RDM Questionnaire

Object of study: The object of study is the RDM process at X.

Purpose: The purpose is to measure any impact of business requirements quality changes on strategic project tasks at X.

Quality focus: The quality focus is on X's employees' perception of the RDM process, the project's success, and X.

Perspective: The perspective is taken from participants' personal opinions of X.

Context: The RDM Questionnaire asked participants to choose a recent strategic project to answer questions on that project's strategic project tasks. Questions asked on a participant's chosen strategic project included the quality of X's RDM process used for the strategic project, the outcome of the strategic project, and the quality of the strategic project delivered. The RDM Questionnaire was designed to elicit the data necessary to answer whether or not a higher RDMMM level for developing strategic projects leads to higher project success.

Section 9.1 shows the actual text of the RDM Questionnaire based on Ellis's original work. The RDM Questionnaire Table 1 is titled "Factors Related to RDM Quality", and it contains twelve questions. The RDM Questionnaire Table 2 is titled "Factors Related to Organization", and it contains nine questions. Each question was designed to focus on a company's projects' success within each category, with the RDM-quality questions relating to good RDM processes and the organizational factors questions relating to participants' perspectives toward the RDM process. The RDM Questionnaire questions in Table 1 were presented in a five-point Likert scale ranging from Very Low skill level to Exceptionally High skill level. The RDM Questionnaire questions in Table 2 were presented in a five-point Likert scale ranging from "Strongly Disagree" to "Strongly Agree".

Three medians were taken:

- for all questions for all participants
- for all questions for co-op students
- for all questions for full-time employees

Also, the difference between co-op students and full-time employees was computed.

The median is taken from all questions in the RDM Questionnaire from Section 9.1 Table 1 and Table 2 to obtain X's RDMMM level.

Table 3.3.2.1.1 shows this case study's relation between the RDMMM levels, RDMMM level definition and RDM Questionnaire's median results. RDMMM Level 1, meaning "very low ability to accomplish good RDM", and RDMMM Level 5, meaning "very high ability to accomplish good RDM".

Table 3.3.2.1.1: RDMMM Level in Relation to RDM Questionnaire Results

RDMMM Level	Ability to accomplish good RDM	RDM Questionnaire Median
1	Very low	1
2	Low	2
3	Neutral	3
4	High	4
5	Very high	5

I concluded that this case study's sample size was too small to accommodate removing any data points.

3.3.2.2 Performance Metrics Performance Metric Data Table

Six standard performance metrics were used in the Performance Metric Data Table: error density, software development productivity, percentage of rework time, cycle time for completing a typical software project, schedule fidelity, and error detection effectiveness [20]. Return on investment was not investigated as the product was still being marketed, and I did not have access to sales records. I conducted informal interviews with X's employees for their professional opinions of how successful they perceived their projects. Interviews focused on discussions of their projects' efficiency, their projects' impact on X's customers, and their projects' preparation for the future. Interview participants were asked at the end of the interview to rate their perceived project success based on interview topics on a Likert scale from 1 to 5 with, 1 meaning, "the project is unsuccessful and produces negative effects on X", to 5, meaning "the project is highly successful and produces significant desired effects on X".

Error Density

I measured a project's error density by the number of errors found per thousand lines of code (KLOC). To count the number or errors found per thousand lines of code, I chose a point in time to measure how many lines of code were in a project and counted the number of active tasks inside of X's Jira database for the project that were reported as errors. Error density calculations relied heavily on the accuracy and completeness of task reports in X's Jira database.

Productivity
I measured a project's productivity by LOC/SM (lines of code written by staff per day). I created a script to count the number of lines of code in a project. This script can be found in Section 10.1. I attempted to exclude all code not written by X's employees, for example, code imported from third-party libraries. I used a script to count how many lines of code were written over an approximate three month period and divided it by the number of staff days in that period.

Rework Time
I measured a project's rework time by dividing the number of days per month used for refactoring by the number of workdays in a month.

Cycle Time
I gathered a set of strategic project tasks completed within a month of when the data was collected and compared its time to complete to strategic project tasks that were similar in size and nature around three months prior. Three months were chosen as the interval because X did quarterly project status reports.

Schedule Fidelity
I measured a project's schedule fidelity by the percentage of projects and milestones completed on or ahead of schedule.

Error Detection Effectiveness
I measured a project's error detection effectiveness, by dividing the total software errors detected during the development stage, by the total errors detected during the project duration, including errors detected during the project software operations stage.

Perceived Project Success
I measured a project's perceived project success by assigning a value on a Likert scale from 1 to 5, with 1, meaning "not successful", to 5, meaning "very successful." A project's perceived project success value was based on information gathered, including, asking X's employees how successful they perceive their project, participating in project meetings, investigating project codebases, and conversing with X's employees regularly about their projects.

Goal Definition: The Performance Metric Data Table
Object of study: The object of study is the developer performance at X.

Purpose: The purpose is to measure the outcome of the strategic project at X quantitatively.

Quality focus: The quality focus is on the developer's performance and the success of the project.

Perspective: The perspective is taken from my personal opinion of X.

Context: A Performance Metric Data Table was filled out for each chosen project. Each chosen project had multiple employees working on the project. I took performance metrics to prevent biased opinions in the RDM Questionnaire based on performance results. Error density, rework time, cycle time, schedule fidelity, and error detection effectiveness metrics were gathered from X's Jira database. Project productivity was measured by a script I wrote (see Section 10.1). I assigned perceived project success based on my personal opinion of a project.

Challenges: I experienced challenges in gathering accurate performance metrics because not all information was recorded correctly. For example, the existence of some critical bugs was not recorded in X's Jira database.

3.3.2.3 CMMI Assessment

In order to reinforce this case study's validity of a CMMI level that I assigned to X, I mandated that the CMMI level can never be higher than the CMM level. For example, if a company is CMM Level 2 then CMMI Level 1 or 2 can be assigned to the company.

A CMMI-AM was conducted in 2019 to confidently assess that CMMI Level 2 had been accomplished between 2017 and 2019. Galin performance metrics from in Section 3.3.2.2 were taken to affirm that a CMMLT occurred between 2017 and 2019.

4 Case Study Findings

Section 4 reviews my collected data on X, including RDM questionnaire results, performance metrics results, CMM findings, and CMMI findings.

4.1 RDM Questionnaire

Section 4.1 reviews data from the RDM questionnaires conducted in 2017 and 2019 by participants at X. Each table found in Section 4.1.1 and Section 4.1.2 comprises 4 columns: total, co-op students, full-time employees, and difference. Each table is representative of a specific time that the RDM Questionnaire was conducted. Each table's column: total, co-op students, and full-time employees, represents a median of participants' responses to one question in the RDM Questionnaire. In each table, each row's difference column value is calculated by taking the row's value in the table's co-op students column minus the row's value in the table's full-time employees column.

4.1.1 RDM Questionnaire Results 2017

Section 4.1.1's tables show data from the RDM Questionnaire conducted in 2017 by participants at X. A valuation of 5 in any column is the best score.

Table 4.1.1.1: "Factors Related to RDM Quality" Median Results from RDM Questionnaire in 2017

Factors Related to RDM Quality	Total	Co-op Students	Full-time Employees	Co-op minus Full-time
Describing project goals and objectives in concise, clear and unambiguous terms	4	4	3	1
Facilitating cross-functional group sessions where requirements were discovered	4	4	3	1
Conducting efficient meetings and marking effective use of stakeholder time	2.5	4	2	2
Accurately documenting the business process behind the application	3	3	2	1
Describing the information flow and key data needed by users at any given time in the business process	3	4	3	1
Assuming that the scope of the project neither significantly changed nor had major in-scope functions moved to follow-on the phases of the project	4	4	1	3
Uncovering project interdependencies or issues that need investigation	4	4	4	0
Cleary describing project risks and assumptions	4	4	4	0
Presenting the results of analysis in clear, well-organized, and readable documentations	2.5	3	2	1
Assessing change requests: determining the impact of these on scope and cost, and the impact of systems changes on business processes	4	4	4	0
Achieving consensus on requirements among stakeholders	3	3	3	0
Getting requirements documented in short, concentrated period of time	3	4	2	2

Table 4.1.1.2: "Factors Related to Organization" Median Results from RDM Questionnaire in 2017

Factors Related to Organization Questions	Total	Co-op Students	Full-time Employees	Co-op minus Full-time
Our organization has formalized approach to doing business requirements which is consistently followed by business analysts on projects	3	3	2	1
Our organization has defined standards for business requirements documentation quality, and assess the work of the analysts against these standards on projects	2	2	2	0
Our organization treats business analysis as a profession and has trained staff dedicated to this function	3	3	2	1
Our organization can predict how much stakeholder time will be needed, and which stakeholders will be involved in the requirements phase of a project	3	3	3	0
Business requirements are traceable, and well integrated into testing at our organization	4	4	3	1
Stakeholders feel that the process of extracting and documenting requirements is efficient at our organization	3	3	2	1
Our organization is excellent at transitioning requirements from business departments into the information technology department	3	3	2	1
I believe the automated business analysis tools that we currently have in place are excellent at helping us elicit requirements	3	4	2	2
I believe the automated business analysis tools that we currently have in place are excellent at helping us manage requirements and requirements change	3	4	2	2

Table 4.1.1.3: Median of All Questions Represented in Table 4.1.1.1 and Table 4.1.1.2 Representing RDMMM levels in 2017

	Total	Co-op Students	Full-time Employees	Co-op minus Full-time
Median	3	4	2	2

Table 4.1.1.3 shows that in 2017, co-op students' perceptions of X's RDMMM Level to be 2 levels higher than full-time employees' perception. I consider co-op students' higher perceptions of X's RDMMM level to be substantially different from full-time employees' perceptions. This difference in perceptions of X's RDMMM level is discussed in Section 4.1.3.

4.1.2 RDM Questionnaire Results 2019

Section 4.1.2's tables show data from the RDM Questionnaire conducted in 2019 by participants at X.

Table 4.1.2.1: "Factors Related to RDM Quality" Median Results from RDM Questionnaire in 2019

Factors Related to RDM Quality	Total	Co-op Students	Full-time Employees	Co-op minus Full-time
Describing project goals and objectives in concise, clear and unambiguous terms	4	4	4	0
Facilitating cross-functional group sessions where requirements were discovered	3	4	3	1
Conducting efficient meetings and marking effective use of stakeholder time	3	3	3	0
Accurately documenting the business process behind the application	2	2.5	2	0.5
Describing the information flow and key data needed by users at any given time in the business process	3	3	3	0
Assuming that the scope of the project neither significantly changed nor had major in-scope functions moved to follow-on the phases of the project	3	4	3	1
Uncovering project interdependencies or issues that need investigation	4	4	4	0
Cleary describing project risks and assumptions	3	3.5	3	0.5
Presenting the results of analysis in clear, well-organized, and readable documentations	2.5	3	2	1
Assessing change requests: determining the impact of these on scope and cost, and the impact of systems changes on business processes	3	3.5	3	0.5
Achieving consensus on requirements among stakeholders	3	3.5	3	0.5
Getting requirements documented in short, concentrated period of time	3	3	2.5	0.5

Table 4.1.2.2: "Factors Related to Organization" Median Results from RDM Questionnaire in 2017

Factors Related to Organization Questions	Total	Co-op Students	Full-time Employees	Co-op minus Full-time
Our organization has formalized approach to doing business requirements which is consistently followed by business analysts on projects	3	2.5	3	-0.5
Our organization has defined standards for business requirements documentation quality, and assess the work of the analysts against these standards on projects	3	3	2.5	0.5
Our organization treats business analysis as a profession and has trained staff dedicated to this function	3	3	2.5	0.5
Our organization can predict how much stakeholder time will be needed, and which stakeholders will be involved in the requirements phase of a project	3	3	3	0
Business requirements are traceable, and well integrated into testing at our organization	3	3	2.5	0.5
Stakeholders feel that the process of extracting and documenting requirements is efficient at our organization	3	3	2.5	0.5
Our organization is excellent at transitioning requirements from business departments into the information technology department	3	3	3	0
I believe the automated business analysis tools that we currently have in place are excellent at helping us elicit requirements	2	2.5	2	0.5
I believe the automated business analysis tools that we currently have in place are excellent at helping us manage requirements and requirements change	2	3	2	1

Table 4.1.2.3: Median of All Questions Represented in Table 4.1.2.1 and Table 4.1.2.2 Representing RDMMM Level

	Total	Co-op Students	Full-time Employees	Co-op minus Full-time
Median	3	3	3	0

Table 4.1.2.3 shows that in 2019 co-op students and full-time employees have the same perception of X's RDMMM level.

Participants' Perceptions of X's RDMMM Level Filtered by Who Provided them with Project Requirements

In 2019, an additional question was added to the RDM Questionnaire, asking participants to indicate who provided them with project requirements. Answers from this additional question show three requirement sources: product line managers (PLM), product managers (PM), and co-op mentors.

I looked at 3 specific groups:

- Co-op students who received requirements from co-op mentors
- Co-op students who received requirements from PMs
- Full-time employees who received requirements from PMs

It is essential to note that a co-op student that received requirements from a PM was not assigned to a co-op mentor. I filtered the data by co-op-students-with-a-mentor and co-op-students-without-a-mentor, because co-op student mentors were found altering requirements, and I wanted to compare co-op students' perceptions of X's RDMMM level with and without the altered requirements.

Figure 4.1.2.4: Participants' Perception of X's RDMMM Levels by Co-op Students Who Receive Requirements from PMs, Co-op Students Who Receive Requirements from Co-op Mentors, and Full-time Developers Who Receive Requirements from PMs in 2019

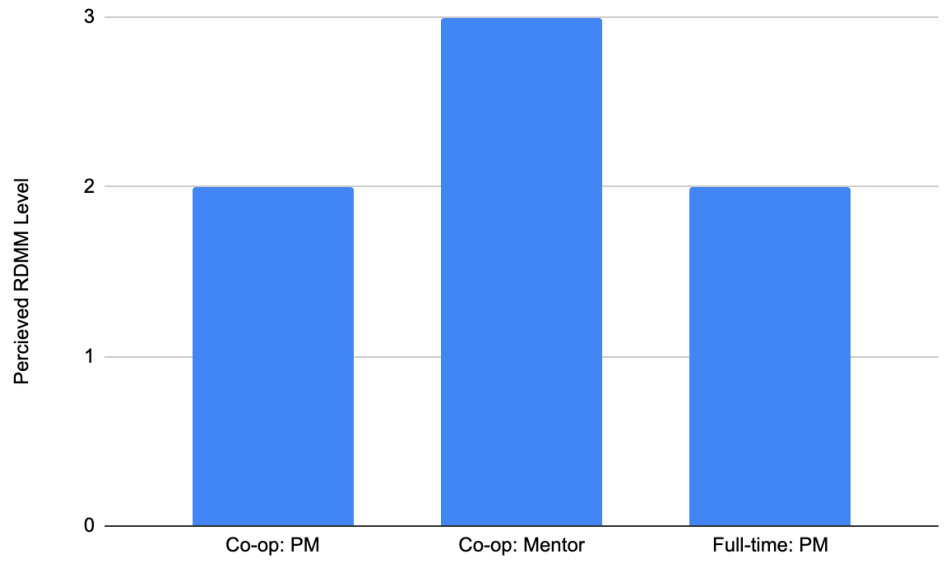


Figure 4.1.2.4 shows that co-op students' perceptions of X's RDMMM level filtered by those who receive requirements from co-op mentors is considerably higher than co-op students and full-time employees who receive requirements from PMs.

In my opinion, this higher perception could be because co-op mentors and co-op students worked on similar tasks. Working on similar tasks allows for a clearer understanding of responsibilities and criteria needed for completing a requirement. In my opinion, X's PMs did not appreciate how important detailed requirements were to translate requirements into code. I discovered that co-op mentors spent time altering requirements from PMs with additional information and clarity before giving a requirement to a co-op student.

Examining X's RDMMM Level with Senior Co-op Students

In the Fall of 2019, I asked three senior co-op students to fill out the RDM questionnaire because I wanted to see if experience could be a suitable substitute for a good RDM.

Figure 4.1.2.5: Participants' Perceptions of X's RDMMM Level by Co-op students and Senior Co-op Students Who Receive Requirements from PMs in 2019

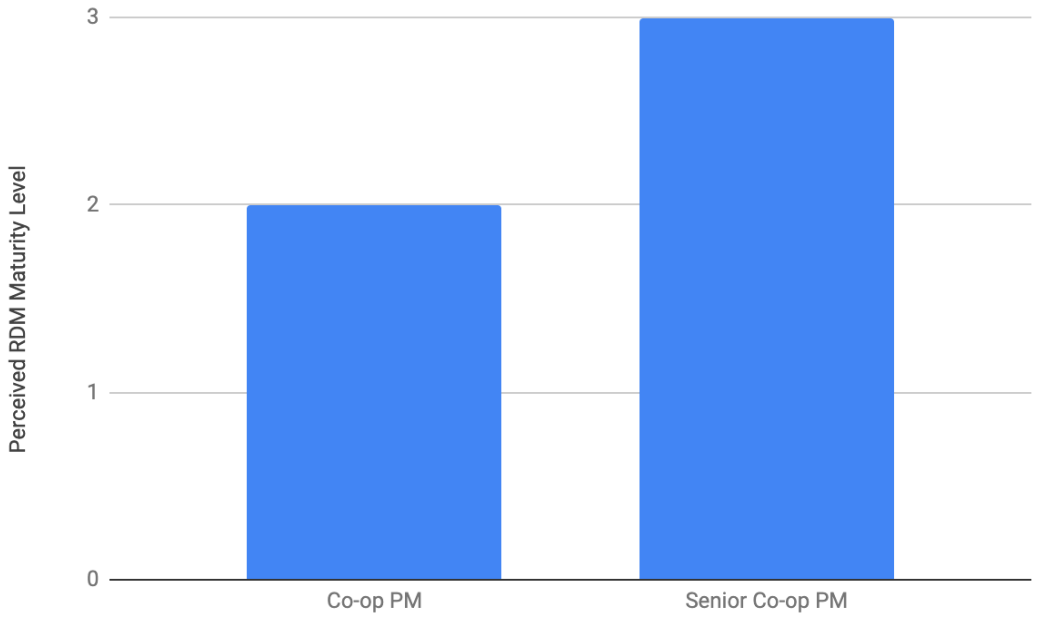


Figure 4.1.2.5 shows data from 2019. Figure 4.1.2.5 shows that senior co-op students who receive requirements from PMs perceive X's RDMMM Level to be 1 level higher than do co-op students who receive requirements from PMs.

I believe that the results from Figure 4.1.2.5 shows anecdotal evidence that experience can supplement a good RDM, because senior co-op students perceive X's RDMMM Level to be 1 level higher than do co-op students who receive requirements from PMs. This evidence shows that experience helped X's employees interpret requirements given by PMs, but a point of RDM is to make a project's performance independent of personal experience.

4.1.3 Determining X's Actual RDMMM Level in 2017 and 2019

Section 4.1.3 aims to determine X's actual RDMMM level in 2017 and 2019 by reviewing the RDM Questionnaire results.

Table 4.1.3.1 RDM Questionnaire Median Scores of All Participants

Year	Perceived RDMMM level
2017	3
2019	3

Table 4.1.3.1 shows the median of all participants' answers in the RDM Questionnaire for 2017 and 2019. Despite X's improvement efforts, Table 4.1.3.1 shows that X had no change in RDMMM level between 2017 and 2019.

I believe full-time employees were more accurate at determining X's actual RDMMM level than co-op students. Due to co-op students' and full-time employees' different perceptions of X, I chose to break the data into two sections: co-op students' perceptions, and full-time employees' perceptions.

Analyzing Full-time Employees Perception of X's RDMMM Level

The following section looks at full-time employees' responses to the RDM questionnaire.

Table 4.1.3.2 RDM Questionnaire Median Scores of Full-time Employees

Year	Perceived RDMMM level
2017	2
2019	3

Table 4.1.3.2 shows that full-time employees' perceptions of X's RDMMM level increased by one level from Level 2 in 2017 to Level 3 in 2019.

Analyzing Participants Results Who Participated in the RDM Questionnaire in 2017 and 2019

Three full-time employees at X, referred to as, "repeat full-time employees", participated in the RDM Questionnaire in 2017 and 2019. These participants' opinions are crucial because they witnessed X's improvement efforts during the entire period from 2017 to 2019.

Table 4.1.3.3 RDM Questionnaire Median Scores of Repeat Full-time Employees

Year	Perceived RDMMM level
2017	2
2019	3

Table 4.1.3.3 shows that the repeat full-time employees' perceptions of X's RDMMM level increased by one level from Level 2 in 2017 to Level 3 in 2019. Table 4.1.3.3 shows the same results found in Table 4.1.3.2.

Analyzing Co-op Students' Perceptions of X's RDMMM Level

Tables 4.1.1.3 and 4.1.2.3 show that co-op students' perceptions of X's RDMMM Level in 2017 is Level 4 and in 2019 is Level 3, a decrease of one level. This decrease is important because X strived to improve its performance between 2017 and 2019, which I expected to result in a higher perceived RDMMM level. This decrease could have resulted because co-ops in 2017 had more experienced co-op mentors than did co-ops in 2019. In my opinion, more experience allows someone to assess more accurately X's actual RDMMM level, explaining the difference in perception between co-op students in 2017 and 2019. In 2017, X's co-op mentors were senior developers, each having over ten years of combined experience at X and prior companies. In 2019, X's co-op mentors changed to a mixture of associate developers and staff developers who typically had substantially fewer years of combined experience at X, and other companies than previous co-op mentors. This decrease could have also resulted from a time discrepancy since co-op students in 2019 were there 2 months longer than co-op students in 2017 by the time the RDM Questionnaire was given to them. In 2017, the RDM Questionnaire was given to co-op students at the start of their co-op term at X. In 2019, the RDM Questionnaire was given to co-op students at the end of their co-op term at X. As co-op students gained more experience they received less help from co-op mentors. This decrease could have also resulted because co-op students in 2017 were given more well-defined remedial tasks than co-op students in 2019.

I believe full-time employees have better understandings of X's actual RDMMM level than co-op students. Tables 4.1.1.3 and 4.1.2.3 show that full-time employees' perceptions of X's RDMMM Level in 2017 is Level 2 and in 2019 is Level 3. The difference in perception of X's RDMMM level between co-op students and full-time employees is two levels in 2017, and is zero levels in 2019. Co-op students in 2019 had perceptions of X's RDMMM level substantially closer to those of full-time employees than did co-op students in 2017. I believe that as co-op students at X gained more experience and were given more complex tasks, the more apparent X's actual RDMMM level became to the co-op students.

X's RDMMM Level Conclusion

Tables 4.1.3.2 and 4.1.3.3 show that the perceptions of X's RDMMM level increased from RDMMM Level 2 to RDMMM Level 3. Tables 4.1.1.3 and 4.1.2.3 show that co-op students' perceptions of X's RDMMM Level in 2017 is Level 4 and in 2019 is Level 3.

The following is a discussion of possible reasons behind why full-time employees are better at accessing X's actual RDMMM level than are co-op students:

1. The average length of time full-time employees worked at X is substantially higher than that of co-op students. Full-time employees' average time spent working at X is over three years in each of 2017, and 2019. Co-op students' average time spent working at X is under one year for co-op students in each of 2017 and 2019.
2. In 2017, out of all participants in the RDM Questionnaire, the highest-ranking participant was a senior developer, X's second-highest-ranking job. In 2019, out of all participants in the RDM Questionnaire, the highest-ranking participant was an architect, X's highest-ranking job. The higher the job ranking, the more an employee is expected to be involved in X's projects' depths and scopes. Senior developers and architects are at least three job ranks higher than co-op students.
3. Due to co-op students' short-term employment, their time to be exposed to many of X's projects and time to thoroughly understand X's software development processes is limited. Co-op students are unlikely to be involved in a strategic project from start to finish.
4. In general, the cheaper co-op students at X had little or no experience with other companies, because they were mostly first term co-op students and, as a result, had little to no reference points for evaluating a company's RDMMM level. Full-time employees are more likely to have had experience at other companies allowing for more reference points when evaluating a company's RDMMM level.
5. In general, co-op students have little or no experience working on different projects and, as a result, have little to no reference points for evaluating a company's RDMMM level. Full-time employees are more likely to have worked on different projects, allowing for more reference points when evaluating a company's RDMMM level.
6. A co-op student typically does not interact with a wide range of people in various positions or teams, limiting the student's access to information.
7. The average number of years a full-time employee worked at X in 2017 is 3.9 years and 4.2 years in 2019. The average number of years a co-op student worked at X in 2017 is 0.15 years and 0.46 years in 2019. More years of experience working at X allows for more exposure to more areas at X, thus allowing for a more accurate perception of X's actual RDMMM level.
8. Table 4.1.1.3 shows that full-time employees' perceptions are two RDMMM levels lower than co-op students'. This difference gives evidence that full-time employees' perceptions and co-op students' perceptions of X's RDMMM level are substantially different.
9. Co-op mentors altered requirements before giving requirements to co-op students. Requirements given to co-op students were of a higher quality than those given to full-time employees.

Therefore, I conclude that full-time employees' perceptions are more accurate when determining X's actual RDMMM level than co-op students' perceptions. I believe X's actual RDMMM Level increased from Level 2 in 2017 to Level 3 in 2019.

4.2 Performance Metric Data Table

Performance metrics were taken at X in order to evaluate X's process improvement efforts.

I decided to list each project's perception of X's RDMMM level from each project's lead developer's RDM Questionnaire results. This decision was because there was an insufficient number of participants representing each project. In addition, I believe each project's lead developer had the most accurate perception of X's actual RDMMM level.

4.2.1 Performance Metric Data Table Results for 2017

Only one project's performance metrics were recorded in 2017 because this case study's scope in 2017 was limited to one project.

Table 4.2.1.1: Project Performance Metrics 2017

Project	Error Density	Productivity	Rework	Cycle Time	Schedule Fidelity	Error Detection Effect-iveness	Project Success	RDM
Octopus	60-80 KLOC	16 lines / day	33%	0%	0%	97.2%	3	2

Table 4.2.1.1 shows Project Octopus's performance metric results in 2017. In 2017, Project Octopus consisted of two employees, a full-time senior developer, and a part-time associate developer.

4.2.2 Performance Metric Data Table Results for 2019

In 2019, performance metric data was expanded to include three other projects along with Project Octopus. The team compositions are as follows,

- Project Octopus consisted of two employees: a full-time associate developer, and a co-op student
- Project Moose consisted of two employees: a full-time architect, and a full-time staff employee
- Project Crocodile consisted of two employees: a full-time staff employee, and a co-op student
- Project Jellyfish consisted of two employees: a full-time staff employee, and a full-time staff employee on contract

Table 4.2.2.1: Project Performance Metrics 2019

Project	Error Density	Productivity	Rework	Cycle Time	Schedule Fidelity	Error Detection Effect-iveness	Project Success	RDM
Octopus	100-150 KLOC	9 lines / day	20%	0%	33%	70%	2	3
Moose	5 KLOC	250 lines / day	50%	33%	100%	90%	5	3
Crocodile	1 KLOC	60 lines / day	40%	20%	75%	100%	4	3
Jellyfish	150 KLOC	75 lines / day	20%	0%	60%	80%	3	3

Table 4.2.2.1 shows the performance metric results of Project Octopus, Project Moose, Project Crocodile, and Project Jellyfish in 2019.

Table 4.2.2.2 2019 Projects Ordered by Best to Worst in Each Performance Metric

	1st place	2nd place	3rd place	4rth place
Project success	Moose	Crocodile	Jellyfish Octopus 2.0	
Error density	Moose	Jellyfish	Crocodile	Octopus 2.0
Lines of code per day	Octopus 2.0 Jellyfish	Crocodile	Moose	
Rework time	Moose	Crocodile	Jellyfish Octopus 2.0	
Cycle time for completion	Moose	Crocodile	Jellyfish	Octopus 2.0
Error detection effectiveness	Crocodile	Moose	Jellyfish	Octopus 2.0

Table 4.2.2.2 sorts projects Moose, Crocodile, Jellyfish and Octopus from most performant to least performant in each performance metric: project success, error density, lines of code written per staff day, rework time, cycle time for completion, schedule fidelity, and error detection effectiveness.

4.2.3 Performance Metric Data For Project Octopus

Project Octopus's performance decreased from 2017 to 2019. In 2019, Project Octopus's error density was higher whereas productivity, error detection effectiveness, and perceived project success was lower than in 2017. In 2019, performance metric data was expanded to include three other projects along with Project Octopus to examine why Project Octopus's performance decreased despite X's improvement efforts.

Project Octopus's lead developer's perception of X's RDMMM Level in 2017 is Level 2, progressing to Level 3 in 2019, an increase of one level. I consider this increase to be substantial enough to be used to draw conclusions on Project Octopus's developer performance changes between 2017 and 2019.

X made substantial improvement efforts to improve developer performance from 2017 to 2019. It is important to note that Project Octopus's team lead was not the same person in 2017 as in 2019. Project Octopus's developer performance was higher in 2017 than in 2019 despite the Project Octopus's team lead's perception of X's RDMMM level in 2019 being higher than the Project Octopus's team lead's perception of X's RDMMM level in 2017. Possible explanations for this decrease in developer performance despite the increase in perceived RDMMM level on Project Octopus are:

1. In 2017, Project Octopus's team consisted of a full-time second-highest ranking senior developer and a part-time second-lowest ranking developer associate developer.
2. In 2019, Project Octopus's team consisted of a second-lowest ranking associate developer, and a low-ranking co-op student. The experience level and overall ranking of developers on Project Octopus substantially decreased between 2017 and 2019.
3. Project Octopus's codebase increased by over 15K LOC from 2017 to 2019, which made Project Octopus a more complex codebase to maintain and manage. When a project has more LOC, it requires more work to maintain project integrity and has more components affected by changes in its codebase.
4. An individual's commitment level to a project can vary. Someone who inherits a project may not be as concerned about its success as its original creator. In 2017, Project Octopus's team included its original creator. In 2019, Project Octopus's team no longer included its original creator.
5. Co-op students are not always fully invested in a company, and could negatively affect projects. In 2017, Project Octopus's team did not include any co-op students. In 2019, Project Octopus's team included one co-op student.
6. Participant M was the lead developer on Project Octopus in 2017. Participant M in 2017 was senior developer, and in 2019 was promoted to architect. Participant M's quality of work on Project Octopus in 2017 contributed to participant M's promotion to the highest ranking job at X.

I believe Project Octopus's team lead in 2017 and Project Octopus's team lead in 2019 perception of X's RDMMM level is accurately assessed based on the data collected and discussed. I believe the decreased developer performance on Project Octopus was because of the extreme difference in experience between Project Octopus's team lead in 2017 and in 2019. In order to ascertain that a developer's performance is not based on his or her experience, a better comparison would be comparing the developer to him or herself at different RDMMM levels.

4.2.4 Performance Metrics by Participant M

It would have been ideal for consistency if Participant M had remained as lead on Project Octopus from 2017 to 2019. However, Participant M became the lead developer on Project Moose in 2019, which made a direct comparison difficult. From Table 4.2.2.1, I compared Project Octopus's performance metrics from 2017 to Project Moose's performance metrics from 2019 because they had the same lead developer. I considered Project Moose's performance metrics from 2019 to be substantially better than Project Octopus's performance metrics from 2017 because Project Moose's schedule fidelity is higher, perceived project

success is higher, productivity is higher, and error density is lower than Project Octopus. Participant M's perception of X's RDMMM Level increased from 2 in 2017 to 3 in 2019.

Participant M's performance on Project Moose was substantially better than his performance on Project Octopus, demonstrating anecdotal evidence that higher RDMMM level results in higher downstream developer performance.

Participant M's increased performance metrics could be a result of his gaining more experience from 2017 to 2019. Participant M's perception of X's RDMMM level could have increased because Participant M became more capable of managing X's RDM tasks.

4.2.5 Performance Metric Data Conclusion

Based on Participant M's performance on Project Octopus in 2017 and Project Moose in 2019, I conclude that higher RDMMM level results in higher downstream developer performance. I further conclude that a lead with higher downstream developer performance has a better perception of X's RDMMM level.

4.3 Determining X's CMMI Level

I assessed X's actual CMMI level based on information gathered through interactions at X, including attending meetings, conducting interviews with employees, and examining X's Jira database.

4.3.1 Determining X's CMMI Level Assessment in 2017

I aimed to determine X's actual CMMI level in 2017.

4.3.1.1 X's CMMI Level Overview

Section 2.5 explains that CMMI baseline level starts at Level 1. A company at CMMI Level 1 has processes that lack a specific purpose and appear chaotic.

The following lists CMMI Level 1 KPAs and for each, lists the evidence that a company is performing it:

1. *A company relies on the competence and heroics of its employees rather than a proven process for success.*

X did not effectively schedule tasks. As a result, X often relied on employee heroics to meet deadlines. X developed an internal system by which it would reward employee heroics with a Hero Cookie. A Hero Cookie is given to an employee who made last-minute changes to a project's codebase, to bring the project back into a presentable state. Hero cookies were given out when X was preparing for a large showing of its products to potential customers.

2. *A company produces products that work; however, they often exceed budget and miss project deadlines.*

Table 4.2.1.1 shows that Project Octopus's schedule fidelity was 0% in 2017, indicating that Project Octopus never met any of its scheduled deadlines.

X constantly reprioritized Jira task's sequential order for completion. X prioritized customers' requirement change requests over scheduled deadline goals, which resulted in short-term customer satisfaction, but also resulted in X's missing scheduled deadlines. X created working products; however, they often exceeded their budget, and missed release deadlines because X spent too much time developing low priority requirements, instead of prioritizing a minimum viable product.

3. *A company often over commits, abandons processes in a time of crisis, and cannot reproduce their past success.*

X's tendency to over-commit before a trade show or major release created the constant need for heroic measures. Under pressure, X would abandon its outlined processes preventing repeatable success. X's success depended on individual employee competence and heroics rather than a proven process.

X's tendency to abandon processes in times of crisis was easily seen in the lack of detail in Jira tasks. Requirements examples below taken from X's Jira database show that X's employees failed to outline the requirements clearly, and developers misunderstood the requirements.

Below are examples of unclear requirements taken from X's Jira database in 2017:

- "Should appear differently when severe faults occur or are imminent."
- "Update (Product H) UI to be more user-friendly."
- "Create an offline editor component."
- "Currently X will fail to connect to the Y if there are Unicode IDs such as Japanese characters: バグ."

According to many of X's developers, the examples above of unclear requirements did not have any other helpful information associated with them, which left too much opportunity for misinterpretation. Jira tasks were mainly created with vague overviews. Jira tasks were written down, referred to, and communicated, but the intent was often understood by only the creator of the Jira task.

Conclusion
I believe X was at CMMI Level 1 in 2017 because the above evidence indicates that X suffered from all of CMMI Level 1 key indicators.

4.3.1.2 X's CMM Level 2 KPA Results

I compared X to CMM Level 2 KPAs because the KPAs have requirements to indicate whether a company has achieved CMMI Level 2. I compared X to CMM Level 2 KPAs also to determine X's progress towards CMMI Level 2. The following are the respective KPAs required to achieve CMM Level 2. Each CMM Level 2 KPAs requirement is rated satisfactory or unsatisfactory based on each CMM KPA goal.

The following lists goals for each CMM KPA Level 2, along with evidence of X's progress in completing each goal.

Requirements Management
Goal 1 - System requirements allocated to software are controlled to establish a baseline for software engineering and management use.

PMs were constantly changing project requirements in order to satisfy customer requests. X allowed its customers to have a direct line of communication with its PMs. X's open line of communication with customers was intended to deliver good customer service, but ended up disrupting a project's scheduling. PMs would prioritize customer requirements without consulting X's developers, resulting in developers missing project deadlines. PMs did not control customer requirement requests through an established process for software engineering and management use.

Goal 2 - Software plans, products, and activities are kept consistent with the system requirements allocated to software.

PLMs and PMs communicated system requirements. System requirements were not controlled in a way that allowed a baseline to be established. Communication between X's customers and X did not follow a process. Software plans, products, and activities were inconsistent with X's system requirements allocated to its software.

Requirements Management Goals Assessment
I did not consider that X achieved the requirements management goals satisfactorily. X's communication with customers was good, but X's lack of effective change control resulted in disruptions to project schedules. X did not follow a process to keep system requirements consistent.

Software Project Planning
Goal 1 - Software estimates are documented for use in planning and tracking the software project.

Employees at X had multiple meetings to scope a project at which time they would document estimates for planning and tracking the project. I observed that time estimates for testing and implementing a feature would often be underestimated.

Goal 2 - Software project activities and commitments are planned and documented.

X used Jira for project planning because Jira allows a project's complete development life cycle to be documented inside of Jira. However, I observed that X's project planning documentation consisted of very vague overviews.

Goal 3 - Affected groups and individuals agree to their commitments related to the software project.

Affected groups and individuals agreed to their commitments related to X's software projects during sprint planning meetings. However, after a sprint planning meeting was completed, I observed PMs adding new Jira tasks not agreed to by all affected groups into X's active sprints. I also observed PMs changing Jira task definitions for Jira tasks selected in X's active sprints.

Software Project Planning Goals Assessment

I did not consider that X achieved the software project planning goals satisfactorily. X used Jira functionality to accomplish software project planning goals. However, X's execution of project planning was ineffective. X's project deadlines were missed when PMs added new Jira tasks and changed Jira task definitions.

Software Project Tracking and Oversight

Goal 1 - Actual results and performances are tracked against the software plans.
Clarity: I will be looking to see if X has visual proof of a system for tracking.

Jira provided management with high visibility on all projects reported in Jira. Jira provided managers with graphs and analytics on project progress and project performance.

Goal 2 - Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans.

Jira would send out a notification when a project's planned deadline was not met. X's management would then initiate a review of a project's deadline expectations. Oversight on projects also included daily team meetings, and bi-weekly sprint planning meetings.

X had team leads that would provide project progress updates daily for management to view in Jira. I observed that X's management reacted too quickly to missed deadline notifications, and management would castigate employees for missed deadlines, before an investigation into X's Jira database for information on why a deadline was missed. Employees would have to take extra time out of their day to explain to management that they had noted in X's Jira database why a deadline was missed, for example, because of a power outage.

Goal 3 - Changes to software commitments are agreed to by the affected groups and individuals.

Team leads had to estimate how long Jira tasks would take their team to complete. The team lead's team members would then be responsible for completing the Jira tasks within its estimated time frame.

Software Project Tracking and Oversight Goals Assessment

I did consider that X achieved the software project tracking and oversight goals satisfactorily. All of a project's progress information was available if recorded into X's Jira database, and project oversight was strong from X's management. X can improve by X's management exercising more due diligence with regular reviews of information contained in Jira.

Software Quality Assurance
X employed an extensive quality assurance team. QA members understood X's process for developing software projects. QA member was an integral part of X's development team.

Goal 1 - Software quality assurance activities are planned.

Software quality assurance activities are planned in Jira. QA members had to plan activities for up-coming sprints, and the members had to make Jira task estimates similar to development teams. QA members fell short of making accurate estimates on time to complete requirements because reported issues were often not scoped out with sufficient detail. For example, a testing machine's availability was not tracked, making it difficult to know when testing could start or be completed.

Goal 2 - Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively.

Each project at X had at least one QA member on it. The QA member attended meetings with developers and PMs. The members were expected to become subject matter experts on their project to negotiate requirements properly. The QA members had to objectively consider how their project's requirement decisions would affect other projects.

Goal 3 - Affected groups and individuals are informed of software quality assurance activities and results.

I observed QA members giving management visibility and information on a project's activities and results by performing demonstrations of finished new requirements.

Goal 4 - Noncompliance issues that cannot be resolved within the software project are addressed by senior management.

X's QA team had a procedure available to speak with senior management when a noncompliance issue occurred.

Software Quality Assurance Goals Assessment
I did consider that X achieved the software quality assurance goals satisfactorily. X's QA team could improve by setting up an availability schedule for all X's testing machines.

Software Configuration Management
X's configuration process was full of manual solutions that were not repeatable as each customer wanted a specialized configuration. X had specialized knowledge to accommodate a customer's desire for custom configurations; however, custom configurations were costly, and error-prone. Creating a standard configuration was difficult for X because of the volume of choices available, and the complexity of the choices.

Goal 1 - Software configuration management activities are planned.

I observed that X had a weak support structure for how a project would be managed in the future. X had a data analyst for reviewing results in Jira, but reviews were not planned and

occurred irregularly. A project's priorities were constantly changing as a result of customer demands.

X had a process to help control changes in its software; however, it was primarily made up of manual tests that were extremely expensive to conduct.

Goal 2 - Selected software work products are identified, controlled, and available.
Clarity: Selected software work products are the products developed at the company.

Most of X's software work products were stored in a cloud server, allowing most of X's employees access to identify what products were being developed at X. X controlled project changes through peer reviews, and manual testing. X had an issue with software work products being available to view. X had projects that were readable but not writable for all developers at X. X did not peer-review all projects. X did not control testing machine usage.

Goal 3 - Changes to identified software work products are controlled

A software project's integrity throughout the project's life cycle was maintained through testing. If the project's code failed a test, the failed test would signify that the project's integrity was no longer maintained. Software projects examined in this case study did not have integrated unit tests. X's QA team performed end-to-end tests to maintain the project's integrity, but most tests were manual. In the long-term, manual testing is more expensive than automated testing.

X controlled changes to its software work products through a difference-checker. Changes in the codebases of software work products were controlled on many projects at X through peer reviews; however, peer reviews were often restricted to a few team members.

Goal 4 - Affected groups and individuals are informed of the status and content of software baselines.

Software documentation at X was written when a product was finished. X did not provide documentation for informing affected groups of a product's baseline during development. X did not create documented statuses and content baselines for informing developers how to build new projects.

<u>Software Configuration Management Goals Assessment</u>
I did not consider that X achieved the software configuration management goals satisfactorily. X's project priorities kept changing, configuration management was manual, testing was done manually, and documentation was written late in a project's development life cycle.

Software Subcontract Management
X had two types of contractor groups:

- Employees from Company Z, a contracting company that provided X with guidance and employee options to fill temporary employment positions.
- Co-op students, primarily from the University of Waterloo (UW), who work at X for one academic term at a time.

Goal 1 - The prime contractor selects qualified software subcontractors.

X's primary contractor was Z. Z selected qualified software subcontractors to perform requested development work at X. X later found that contractors selected by Z were not qualified to complete development work to X's satisfaction. Z's contractors often produced unreliable code that needed to be reworked by X's full-time employees.

X selected co-op students themselves through an interview process. Most of the time, X was successful at selecting qualified software subcontractors for co-op positions.

Goal 2 - The prime contractor and the software subcontractor agree to their commitments to each other.
Clarity: I will look for clearly written commitments signed by both parties of the expectations and timelines.

X and Z's contractors signed legal agreements summarizing their commitments to each other. However, these legal agreements often resulted in higher expenses for X. For example, Z would charge X for Z's contractors to fix bugs caused by Z's contractor's faulty work. As another example, X would save money initially by entering into contracts with Z, when Z would become the source code owner. X would then be forced to pay Z for any additional work on the source code. Contracts that made Z the source code owner could encourage Z to produce low-quality code, to perpetually create more work by being paid to alter the source code. Z's contractors did follow the written agreement between them and X; however, they worked in line with the work-to-rule job action. Z's contractors did not complete tasks beyond what they were required to complete, and would follow all regulations precisely. For example, a contractor from Z checked-in source code that broke the codebase, and would not voluntarily extend the workday to fix it.

X and co-op students signed legal agreements summarizing their commitments to each other. Co-op students often worked above and beyond the written agreement.

Goal 3 - The prime contractor and the software subcontractor maintain ongoing communications.

X found it harder to manage Z's contractors than X's co-op students because Z's contractors worked offsite at Z's facility, and X's co-op students worked onsite at X's facility. X also found it harder to manage Z's contractors than X's co-op students, as Z's contractors used a separate Jira database that many of X's full-time employees could not view, and X's co-op students used X's Jira database.

Goal 4 - The prime contractor tracks the software subcontractor's actual results and performance against its commitments.

X found that Z's contractors and X's co-op students both produced low-quality work. X anticipated that its co-op students would produce low-quality work because of co-op students' lack of experience. However, X still hired co-op students because they were less expensive, and over time they gained enough experience to do an adequate job. X used UW's

co-op program as an opportunity to observe the work habits of potential candidates to hire after graduation. X expected high-quality work from Z's contractors because Z's contractors had experience, were highly recommended, and were considerably more expensive than co-op students. After X tracked Z's contractors' actual results and performance, X was unsatisfied with Z's contractors' quality of work.

Software Subcontract Management Goals Assessment
I did consider that X achieved the software subcontract management goals satisfactorily with their co-op students. I did not consider that X achieved the software contract management goals satisfactorily with their contractors from Z. X can improve Z's contractors' performances by conducting interviews before hiring Z's contractors, requiring Z's contractors to work onsite at X's facility, and having Z's contractors use X's Jira database.

4.3.1.3 X's CMM Level 2 KPA Conclusion

In 2017, X did not accomplish all of CMM Level 2 KPAs to my satisfaction; thus, X did not achieve CMM Level 2. I did not consider X for CMMI Level 2 because X failed to reach CMM Level 2.

4.3.1.4 X's CMMI Level Conclusion

In 2017, I concluded that X was at CMMI Level 1 because all of X's CMM Level 2 KPAs were not accomplished to my satisfaction, and X fits the CMMI Level 1 key indicator description.

4.3.2 Determining X's CMMI Level Assessment 2019

I aimed to determine X's actual CMMI level in 2019.

4.3.2.1 X's CMMI Level Overview

By 2019, X no longer appeared to function chaotically and without purpose. X showed signs of having a planned, performed, measured, and controlled process at a project level.

The following lists CMMI Level 2 key indicators along with evidence if X suffers from each key indicator:

1. *A company's projects are managed by a process that is documented.*

I observed that X established a requirements process following a planned, performed, measured, and controlled workflow. X used Jira for project management, a process for creating consistent Jira stories was instituted, and workflows were documented. X's documented development cycle workflow included: "To Do" -> "In Progress" -> "Ready For Test" -> "In Test" -> "Ready For Review" -> "Done".

2. *A company utilizes the process in place during times of stress.*

X maintained their process during times of stress; however, developers were still working excessive hours, indicating that the process needed refinement. Though the process stayed consistent, times of stress frequently occurred at X because X would overcommit to complete customer requirements within short timelines.

3. *A company's project's status is visible to management at defined points in the project's development life cycle.*

X utilized Jira in a way that every employee can view a project's status within every stage of a project's development life cycle. X used Jira's roadmaps to see a project's status at defined points of a project's development life cycle.

4. *A company's commitments are established with relevant customers and are revised as needed in a change-controlled process.*

In 2017, X allowed customers to make many changes without penalty. By 2019, X began to push back against customers within a change-control process. X began to state its commitments and requirements more clearly in its customer contracts. X started a change control process for customer requests that needed to be approved by PMs and developers.

Although X made strides towards integrating more with one of X's sister branches, I do not believe X made a strong enough effort to warrant investigating CMMI Level 3's key indicators.

Conclusion
I believe X is at CMMI Level 2 because the above evidence showed that X accomplished all CMMI Level 2 key indicators.

4.3.2.2 X's CMM Level 2 KPA Results

The following lists goals for each CMM KPA Level 2, along with evidence of X's progress in completing each goal.

Requirements Management
From 2017 to 2019, X increased their use of Jira to manage customer requirements. X attempted new approaches using Jira, with mixed results.

When X first started using Jira in 2017, its use by employees was low because X's employees had bad experiences with other issue tracking software; however, from 2017 to 2019, I observed that X's requirement management process improved in correlation with higher Jira use.

Goal 1 - System requirements allocated to software are controlled to establish a baseline for software engineering and management use.

X's customers still had a direct line to communicate to PMs; however, more rules around adding, removing, or changing requirements were in place. When PMs attempted to add, remove or change requirements, there was resistance from X's team leads.

X focused on establishing an improved baseline and developing a minimal viable product for many of their new projects in 2018 and 2019.

Goal 2 - Software plans, products, and activities are kept consistent with the system requirements allocated to software.

Jira allowed developers to become more involved in X's requirements process. Instead of passing individual files that represented parts of X's requirements process, which caused files to become out of date quickly, Jira would host the most recent version of X's requirements process files so that anyone with access could view it. Jira had built-in requirements tools, which X's management initially resisted following. For example, X's management did not believe Jira could convey a high enough urgency for some requirements. X's management decided to create a system outside of Jira for their desired workflow. This system gave managers a priority card that they could use to prioritize completing a requirement above any requirements in Jira. X's management thought the priority card system would prevent massive undesirable outcomes. However, X's management's unrestricted use of priority cards resulted in disputes between managers, it interrupted X's developer's workflow, and requirements in Jira were ignored. X's management eventually abandoned the priority card system and used Jira to schedule requirements, reestablishing traceability and accountability.

Requirements Management Goals Assessment

I did consider that X achieved the requirements management goals satisfactorily. X made dramatic improvements in Requirements Management since 2017 because X increased its use of Jira for requirements management tasks. X experienced negative effects when they used a secondary system outside of Jira that had little traceability and accountability. Due to these negative effects, X's management started to use Jira with more commitment and regularity. X can better improve by implementing tools that continue to enhance their Jira experience.

Software Project Planning

X continued to use Jira for project planning in 2019, allowing decision-makers to see X's teams' project plan.

In 2017, many of X's employees believed that X would have to spend years before releasing a minimum viable product because X's area of expertise was thought to be complicated. I believe that X's product could have been finished within the two years I observed X if X followed proper project planning. Instead, many of X's project plans were constantly derailed.

Goal 1 - Software estimates are documented for use in planning and tracking the software project.

Software estimates were documented for planning and tracking a software project's progress; however, X continued to underestimate completion times. Software estimates were documented using story points. X would compare a project's total story points completed within a sprint against a project's total completed story points during the sprint for tracking and review.

Goal 2 - Software project activities and commitments are planned and documented.

In 2017, X's practice was not to release a project's plan until an entire project was completed, which could take years. By 2019, X's practice had to release project plans every quarter, allowing continuous improvement and earlier customer project review.

Goal 3 - Affected groups and individuals agree to their commitments related to the software project.

Project planning used to be done only by managers, but in 2019 X began to include team leads and developers. Authority was given to team leads and developers to reject tasks until requirements were clear, make estimates on tasks, and be allowed to decide when a task should be scheduled to be complete.

Software Project Planning Goals Assessment
I did consider that X achieved the software project planning goals satisfactorily. X included more team leads and developers in project planning, and X began to acknowledge the value in planning meetings, frequently releasing production-ready projects, weighing in on person-months, and planning future projects.

Software Project Tracking and Oversight
Project tracking and oversight continued to improve with greater use of Jira.

Goal 1 - Actual results and performances are tracked against the software plans.
Clarity: I will be looking to see if X has visual proof of a system for tracking.

Jira was used as a tracking system. X made a serious commitment to using Jira. For example, if an employee failed to update Jira daily, he or she would be reprimanded. This approach might encourage updates to occur, but it might not encourage the updates to be accurate.

Goal 2 - Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans.

X's management took practical actions when they noticed a project deviating from its project plan. Developers would actively take corrective action with each other to alleviate blockers when results were deviating significantly from software plans. Examples of developers' corrective actions often included working overtime, and reorganizing a project's task priorities to put the project back on track.

Goal 3 - Changes to software commitments are agreed to by the affected groups and individuals.

Most of X's employees from X's co-op students to its director of research and development benefited from Jira, which allowed affected groups and individuals the opportunity to agree to software commitment changes.

Software Project Tracking and Oversight Goals Assessment
I did consider that X achieved the software project tracking and oversight goals satisfactorily. Jira was used to provide immediate and universal access to all project information. X could improve by finding positive ways to encourage its employees to update Jira daily.

Software Quality Assurance
QA members continued to function at the same high level that it was functioning at in 2017.

Software Quality Assurance Goals Assessment
I did consider that X achieved the software quality assurance goals satisfactorily. X maintained their high level of success within the Software Quality Assurance goals.

Software Configuration Management
Between 2017 and 2019, X worked hard to move away from manual to automated configuration deployments and testing. X's employees were not satisfied with their initial automation solution's quality for their configuration tooling. The automation solution was not used for most tasks. A generalized configuration tool that would not require a trained technician was hard for X to make. I observed that X made a few attempts to fix its configuration tooling with some success.

Goal 1 - Software configuration management activities are planned.

X created an architecture team assigned to monitor and plan its software projects throughout a project's development life cycle to maintain project integrity.

Goal 2 - Selected software work products are identified, controlled, and available.
Clarity: Selected software work products are the products developed at the company.

All of X's software work products were stored in a cloud server allowing all of X's employees to identify what products were developed at X. By 2019, all of X's work products were accessible to read and write by X's developers. All of X's work products were controlled with manual and automated tests. X started a formal process for peer reviews. Though X had all projects available on X's cloud servers for X's developers, X's managers controlled who was permitted to delete projects on X's cloud servers.

Goal 3 - Changes to identified software work products are controlled.

When a developer committed code to a project, a script would run to display the old code and new code alongside each other. X's developers controlled what code was allowed into production by forcing all new code to pass all tests and be peer-reviewed. Tests were a combination of automated and manual tests.

Goal 4 - Affected groups and individuals are informed of the status and content of software baselines.

X created baseline projects that contained basic code and infrastructure to assist with new project creation. These baseline projects were committed to X's cloud server in a way that

allowed affected groups and individuals to stay informed on X's changing content of X's software baseline.

Software Configuration Management Goals Assessment
I did consider that X achieved the software configuration management goals satisfactorily. X implemented more automated configuration, deployments, and automated tests. X could improve by continuing to move its configuration process into automation.

Software Subcontract Management
Goal 1 - The prime contractor selects qualified software subcontractors.

Initially, X would rely heavily on Z's recommendations for contractors, but X was dissatisfied with Z's contractors' quality of work. X's staffing budget was time-sensitive. If X's managers did not hire someone within an allotted time frame, they would lose their allocated hiring budget. X's managers would hire underqualified candidates in order to meet hiring deadlines. Budget management changes were made to give X more time to interview and vet contractors themselves, resulting in X hiring higher-quality contractors. X did have a recorded formal process on how to conduct an interview; however, it was not highly detailed.

Goal 2 - The prime contractor and the software subcontractor agree to their commitments to each other.
Clarity: I will look for clearly written commitments signed by both parties of each other's expectations and timelines.

X started to hire contractors similar to how X's full-time employees were hired. Contracts between contractors and X were drawn up for a specific amount of time, rather than for a specific task. Contracts were drawn up such that X became a project's source code owner. Contractors attended the same sprint planning meetings as X's employees. Both contractors and X's employees had to agree to project commitments before commitments were assigned.

Goal 3 - The prime contractor and the software subcontractor maintain ongoing communications.

X instituted changes to improve communication with Z's contractors. Previously all communication with Z's contractors had to be done through a 3rd party mediator. X required Z's contractors to work at X's facility, integrate into X's teams, and use X's Jira database.

Goal 4 - The prime contractor tracks the software subcontractor's actual results and performance against its commitments.

Contractors used X's Jira database, and participated in X's requirements management, software project tracking, project oversight, and quality assurance. Contractors were managed like full-time employees, which allowed X to better track contractors' actual results and performance efficiently.

Subcontract Management Goals Assessment

I did consider that X achieved the subcontract management goals satisfactorily. X could further improve by developing a more formal interview process, and by producing a detailed interview process document.

4.3.2.3 X's CMM Level 2 KPA Conclusion

In 2019, I concluded that X was at CMMI Level 2 because X accomplished all of CMM level 2 KPA goals to my satisfaction, and X fits CMMI Level 2 key indicators descriptions. Section 4.3.2.4 reinforces my conclusion that X is at CMMI Level 2 by verifying if a CMMLT occurred and by evaluating X with CMMI-AM.

4.3.2.4 X's CMMI-AM Conclusion

I filled out a CMMI-AM for X [29]. Out of the CMMI-AM 39 questions, I deemed 11 did not apply to X and removed them from the evaluation. My CMMI-AM mode score result for X is 7 in 2019. I assumed a CMMI-AM mode score below 4 would represent a company at CMMI Level 1. I believe X's CMMI-AM mode score was high enough to justify assigning X CMMI Level 2.

4.3.2.5 X's CMMLT Conclusion

Table 4.3.2.5.1 shows the minimum performance improvements for a CMMLT to have taken place from CMMI Level 1 to CMMI Level 2, performance changes for Project Octopus between 2017 and 2019, and performance changes between Project Octopus in 2017 and Project Moose in 2019.

Table 4.3.2.5.1: Comparing Project Performance Changes to CMMLT

	Error Density	Productivity	Rework	Cycle Time	Schedule Fidelity	Error Detection Effectiveness	Project Success
CMMLT CMM 1 to CMM 2 [20]	16-75	25-74	31-51	26-69	33-79	60-80	N/A
Octopus 2017 and Octopus 2019	-70%	+56%	+66%	N/A	0 to 33%	-39%	-1 Level
Octopus 2017 and Moose 2019	+1400%	+1500%	-50%	N/A	0 to 100%	-8%	+2 Level

Table 4.3.2.5.1 shows that both project comparisons were above the median bounds for a CMMLT between CMMI Levels 1 and 2 to have taken place. I conclude that X could have transitioned from CMMI Level 1 to CMMI Level 2 because a CMMLT could have occurred between 2017 and 2019 for the projects I observed at X.

4.3.2.6 X's CMM Level 3 KPA Assessment Results

Since X reached CMM Level 2, the following looks to see if X accomplishes all of CMM Level 3 KPA goals to show X's progress towards CMMI Level 3.

The following lists goals for each CMM KPA Level 3, along with evidence of X's progress in completing each goal.

Organization Process Focus

Goal 1 - Software process development and improvement activities are coordinated across the organization.

I observed that X improved its organization process focus when management initiated a solid push to use Jira to coordinate activities across X. However, I observed that PMs used X's older issue tracking software named Target Process to oversee projects already represented in Jira. X had to maintain the same information within two separate issue-tracking software programs, which increased the cost of managing a project. To keep track of what task each employee was working on, X initiated a practice of having employees sign off on their current work. For example, X initiated a practice that if another employee took on a Jira task, the original owner would sign off, and the receiving employee would sign on, indicating they were currently responsible. This practice helped to identify who was presently responsible for a task.

Goal 2 - The strengths and weaknesses of the software processes used are identified relative to a process standard.
Clarity: I will not consider this goal accomplished until the Organization Process Definition goals below are met

X's software process standard reviews were rare and often only initiated after multiple developers at X complained about them. I did not observe any formal documentation of X's software process standards.

Goal 3 - Organization-level process development and improvement activities are planned.

Despite X's desire to use Jira exclusively for all of its RE processes, X did not have the required expertise to use all of Jira's functionalities adequately. X often misused Jira's functionalities. For example, management wanted employees to record Jira task completion times. In some cases, I observed that X's developers underestimating the time it takes to complete Jira tasks, leading to developers skewing the actual time to complete a Jira task to match the original completion time estimate. Even though developers became more accurate in estimating the time to complete Jira tasks, the estimation functionality in Jira was abandoned because X's employees lacked confidence in the accuracy of recording the completion times. Another example of failed use of Jira functionality occurred when X wanted its project development cycle updated consistently. As individual development cycle steps were completed, they were to be immediately marked in Jira as done; however, employees did not follow through; instead, they waited until the end of a project's development life cycle to mark all development life cycle steps as done. As a result of this misuse of Jira functionality, the daily reviews on the project did not accurately report the actual state of the development cycle.

Organization Process Focus Goals Assessment

X did not achieve all of the Organization Process Focus goals to my satisfaction. I observed that X did not strictly enforce Jira practices as policy. I believe X understood that better RE processes and better software project management processes would improve project success, but X's effort was insufficient to develop and maintain these processes.

Organization Process Definition

Goal 1 - A standard software process for the organization is developed and maintained.

X tried to utilize Jira's existing set of software process management assets. X believed that Jira would provide a way for all employees to participate, provide X with starting guidelines, and help initiate standard software processes throughout X. X customized some of Jira's assets to satisfy its goals. X used Jira to create standardized templates for all employees to follow. For example, X already had a development-life-cycle template before Jira was introduced but representing the template in Jira forced X's employees to follow its prescribed development life cycle unless overridden.

Goal 2 - Information related to the use of the organization's standard software process by the software projects is collected, reviewed, and made available.

X initiated bi-weekly backlog-grooming meetings for each project to discuss, review, and prioritize its Jira backlog. Backlog grooming focuses on managing developer time and short-term goals but does not consider long-term benefits to X.

Organization Process Definition Goals Assessment

X did not achieve all of the Organization Process Definition goals to my satisfaction. I observed that X developed few software process assets or maintained Jira's software process assets. However, X did initiate backlog grooming for planning short-term goals, and X did learn how to use some of Jira's software process assets.

Training Program

Goal 1 - Training activities are planned.
Clarity: A training schedule will be provided.

Team leads at X created and planned training sessions for new team members. In addition, X organized weekly Lunch-and-Learn sessions, with their topics and meeting agendas posted ahead of time.

Goal 2 - Training for developing the skills and knowledge needed to perform software management and technical roles is provided.

X started financing training conducted by third-party companies; however, this training was available for a select number of individuals at X. For example, an entire team at X was not allowed to go to a Systematic Customer Resolution Unraveling Meeting (SCRUM) training despite that all team members needed to understand SCRUM to follow the Agile framework properly. X also failed to set up a process for sharing the SCRUM training information among employees who were not selected to attend or were not available to attend. I observed

that X had a good idea to train its employees, but X did not have a logical process on whom to send for training or what training was needed. For example, X's DevOps team was approved to send co-op students all expenses paid for training on Kubernetes, but co-op students are not employed long enough to add value to X after training is completed.

Goal 3 - Individuals in the software engineering group and software-related groups receive the training necessary to perform their roles

Training sessions included a general overview of the tools and technologies used by X, which went into very project-specific details, including demonstrations of the projects.

X's employees started to initiate Lunch-and-Learn sessions. High attendance to Lunch-and-Learn showed X's employees' desire to learn, but an informal meeting during lunch is a poor substitute for formal training organized by the company.

When X was legally required to train its employees, X strived to meet federal standards. X's managers tracked their teams' training progress and would send reminder emails to employees late on completing training tasks. X still fell short in ensuring that all employees had all legally required training. For example, employees without server-room training were found in X's server rooms unaccompanied by someone with server-room training. X attempted to restrict access to its server rooms by having locks on its server-room doors, but their server-room doors were kept open because the server-room would often overheat.

Training Program Goals Assessment

X did not achieve all of the Training Program goals to my satisfaction. X did commit to training its employees but did not create a logical selection process for which employees were to receive additional training. X failed to set up a shared training process when X's training budget was too small to send all employees for training. I also had serious concerns about how X left its server room doors open.

Integrated Software Management

Goal 1 - The project's defined software process is a tailored version of the organization's standard software process.

Clarity: I will not consider this goal accomplished until the Organization Process Definition goals above are met

X integrated its software engineering and management activities into Jira to allow these activities to be coherent across X. X took time to think about and set up Jira defaults; however, I did not find documentation on why X's defaults were chosen. Most of X's teams used the same templates found in Jira without any customization, which did not meet many of X's teams' needs. For example, many teams ended up with unnecessary development life cycle steps in their workflow that they skipped.

Goal 2 - The project is planned and managed according to the project's defined software process.

I observed that X established a process for requirements following a planned process, but I did not find a defined project-specific software process. Jira was used for project

management, and a process for creating consistent Jira stories and workflows was documented.

Integrated Software Management Goals Assessment
X did not achieve all of the Integrated Software Management goals to my satisfaction. No substantial documentation of integrated software management was found, and X's Jira templates were not customized to meet individual team needs.

Software Product Engineering
Goal 1 - The software engineering tasks are defined, integrated, and consistently performed to produce software.

X recorded software engineering tasks in Jira. However, it was difficult to find a specific software engineering task in Jira, and some employees made little effort to keep Jira task information up to date. As a result, when Project Moose was developed, Project Moose's developers did not utilize aspects of X's Jira database. For example, Project Moose was primarily developed from the personal knowledge of developers.

Goal 2 - Software work products are kept consistent with each other.

X had a User Experience (UX) team dedicated to keeping its work products consistent with each other. X's developers still suffered from not having a cloud repository to effectively share reusable components across X's teams. X's UX team created and maintained a small-scale Node Package Manager (NPM) library for sharing Scalable Vector Graphics (SVG) images. This NPM library saved the UX team from creating duplicate icons and allowed all of X's employees to have the same version of an icon and know what icons had already been created. (See the paragraph about duplicate code in Section 5.2.1.) However, I observed that X's developers did not create a process for maintaining code consistency, and as a result, code consistency was difficult to manage.

Software Product Engineering Goals Assessment
X did achieve all of the Software Product Engineering goals to my satisfaction. X can improve by creating more documentation, thus guarding against losing employees' specific knowledge bases when employees leave.

Intergroup Coordination
X had a social committee dedicated to planning fun events for X's employees. Events would include but were not limited to movie nights, board games, go-karting, or dinners. Outside of company-funded events sponsored by X, X's employees continued to socialize together. For example, X's employees played card games at lunch, met for drinks, or played a casual soccer game together.

Goal 1 - The customer's requirements are agreed to by all affected groups.

In 2017, new customer requirement changes would be prioritized over agreed upon developer tasks, causing delays in accomplishing project deadlines. In 2017, X did not get agreement from all affected groups before accepting a customer requirement change. By 2019, X had initiated a new process for dealing with customer requirement changes. X allowed developers

to review customer requirement contracts before signing contracts with its customers. X improved its customer requirements change control by tracking over-commitments, improving project prioritization, and increasing backlog management. In 2019, X's customer requirement change control process continued to be refined and introduced into X's culture.
.
Goal 2 - The commitments between the engineering groups are agreed to by the affected groups.

X's Project Moose had engineering groups affected by decisions made on Project Moose at X and at its sister branch in Toronto. By 2019, X had improved its communication with its sister branch in Toronto by collaborating on Project Moose. X tried to ensure all affected groups from X and its sister branch in Toronto agreed to commitments through sprint planning meetings.

Goal 3 - The engineering groups identify, track, and resolve intergroup issues.

By 2019, X had started an initiative to place more of its developers physically in off-site locations to work temporarily. For example, X's developers worked at X's sister branches and customer work facilities. X benefited from sending developers physically to various locations because it improved intergroup coordination; however, it was expensive to transport and house developers at off-site locations. For example, it was expensive when X sent a developer to Milan, Italy, for six months. X's customer in Italy preferred a developer on-site because the developer on-site could address concerns immediately, and X's developer was then able to deal with physical hardware configuration issues.

Intergroup Coordination Goals Assessment
X did achieve all of the Intergroup Coordination goals to my satisfaction. X started sending developers to customer locations, resulting in better relationships between customers and X. However, the better relationships came at a high financial cost, namely travel and accommodations. X also initiated social events for its employees to improve its interpersonal working relationships, and keep morale high.

Peer Review
Goal 1 - Peer review activities are planned.

Peer reviews became part of X's process. X's developers and QA members had to accept a task's documented criteria for completion before development on the task commenced. A task's completion criteria consisted of its description, precondition, postcondition, and test cases. A developer had to finish all task criteria before asking a QA member for a peer review.

On occasion, peer reviews took place in real-time through pair-programming. For example, a developer and a QA member would pair-program during task creation, allowing the review to be performed in parallel with development.

Goal 2 - Defects in the software work products are identified and removed.

X learned lessons from its failure to peer-review. For example, X discovered peer-review failures within Project Lion. Project Lion had one developer, and its developer did not allow anyone to ever peer-review or inspect his source code. When Project Lion's developer resigned at X, X's remaining developers gained access to inspect Project Lion's source code. They did not find any notes on deciphering its code. Another example was that some developers at X would lock their codebases, restricting access to their source code. X changed from allowing each employee to lock up his or her own code base to forcing that a project's code bases are accessible to all of its qualified employees.

Peer Review Goals Assessment

X did achieve all of the Peer Review goals to my satisfaction. Peer reviews became part of a documented process, project codebases did not restrict access to qualified employees, and a collaborative working environment became part of X's culture.

4.3.2.7 X's CMM Level 3 KPA Conclusion

In 2019, X did not accomplish all of the CMM Level 3 KPAs to my satisfaction; thus, X did not achieve CMM Level 3. I did not consider X for CMMI Level 3 because X had failed to reach CMM Level 3.

4.3.2.8 X's CMMI Level Conclusion

In 2019, I believe X was at CMMI Level 2 because:

- X accomplished all of the CMM Level 2 KPA goals to my satisfaction
- X fit CMMI Level 2 key indicator descriptions
- X scored high enough on my CMMI-AM evaluation to warrant CMMI Level 2
- Observed projects' performance improvements between 2017 and 2019 were high enough to indicate a CMMLT took place
- X was not considered for CMMI Level 3 because X's did not accomplish all of the CMM Level 3 KPA goals to my satisfaction

5 Discussion

This section discusses conclusions I drew from my observations of X between 2017 and 2019. This section also includes my recommendations to X on how to improve its developer performance, RDMMM level, and CMMI level.

5.1 Investigation in 2017

Overview of the Investigation

I observed several of X's working practices that caused substantial disruption to X's workflow in 2017. Section 5.1.1 discusses each disruptive work practice observed in 2017 under these subheadings: Frequent Customer Requirements Changes; Insufficient Requirements Given to Developer; Contract Penalties; Customer Requirements Taking Priority and Causing Delays; Duplicate Code; Lack of Hardware Impacts Development; Company Culture, and Lack of Documentation.

I made recommendations to X that focused on work practice improvement at X in 2017. Section 5.1.2 discusses each recommendation given in 2017 under these subheadings: Enhancing and Improving the Quality Management System; Improving Status Reports of a Project to its Managers; Research Before Actions; Reducing the Training Time for Co-op Students, and Using CMMI.

5.1.1 Observations 2017

Frequent Customer Requirement Changes

X's competitive advantage in a tight market was its ability to provide customized software products. In order to provide customized software products as a service, X's contracts allowed customers to make requirement changes. However, customer requirement changes created extra work for developers. X's vague, open-ended contracts with its customers allowed X's customers almost unlimited opportunities with regard to customer requirement change requests. X's customers took advantage of these opportunities with frequent customer requirement change requests. Salespeople would frequently approve a customer requirement change request without involvement from X's development team. Salespeople agreed to customer requirement changes in order to complete a sale and earn a signing bonus. However, they did not consider whether the customer change request would cause disruptions to a project or whether it was even possible to complete. Salespeople did not suffer from repercussions if a customer requirement change request they approved was not completed.

I observed that X's developers expressed frustration over the constant customer change requests, demonstrating low morale, and feeling dissatisfied with the quality of work they produced.

Insufficient Requirements Given to Developers

X's parent company did not provide support for salespeople to aid in requirements gathering. Salespeople neglected to detail customer requirements. Salespeople neglected to thoroughly analyze and document the requirements in a way a developer could understand.

Often, when a developer at X received a requirement that was not well defined, to proceed, the developer had to interpret the requirement by him or herself. Misinterpretation of a

requirement can often occur without support from its creator to understand its original intention.

I observed that many of X's developers frequently would complete a requirement in a way that was not consistent with the requirement's original intent. I also observed that many of X's developers frequently neglected to update a requirement with new information to explain their interpretation of the requirement. I further observed X's developers spent excessive amounts of time to interpret requirements, wasting development time.

Contract Penalties

X entered into contracts with customers that stated that X would suffer financial repercussions if X missed a contract deadline. For example, X had contracts that would state financial penalties as high as CAD 80,000 per day if the contract deadline was missed. X's contracts with customers were vague, which allowed customers to make outrageous requirement change requests that X was contractually obligated to fulfill by the original deadline.

I observed that X did not renegotiate new contract deadlines when a customer requirement change request was made, which skewed the original contract's time estimates. As a result, X often missed a contract deadline or had its developers work extreme overtime hours to meet a contract deadline.

Customer Requirements Take Priority

I observed that X competed in a small and very competitive market, which caused a lot of pressure on X to keep customers satisfied. I observed that X would promise customers specialized work in order to win customer contracts. To stay competitive, X allowed frequent unsolicited customer requirement changes, which resulted in X's inability to create a proper change control process. X's projects suffered from constant interruptions when a developer's focus changed from project completion to completion of customer requirement changes.

I observed that X neglected to set up a process to manage its customer requirement change requests. Without a process, I observed that X's developers' work was frequently halted, which was expensive for X because of high development costs.

I observed that X made a customized version of projects for each customer, as no project could be configured by the customer. For example, X would manually hard code a set of colors for how objects should appear on a computer screen for each of X's customers. Alternatively, X could have created a UI from which customers could have chosen a color from a color wheel.

I observed that X's management of individual customer needs led to chaos and contributed to poor code quality. X's developers focused on customer requirement change completions and neglected proper code reviews. X prematurely released incomplete versions of its products for customers to review and elicit feedback from but resulted in additional customer requirement changes.

I observed that X prioritized customer requirement changes over completion of a minimal viable product.

Duplicate Code

I observed that X's developers did not have an effective process to avoid duplicate code. Many of X's projects were placed on a cloud-based server that allowed a project's codebase to be shared amongst X's employees. Access to a project's codebase allows other developers to reference it for their project. X's cloud-based servers did not have the functionality to search quickly for a code segment, causing developers to spend a considerable amount of time to find the code segment they required. In addition, when a code segment required was found, developers would copy and paste the segment from one project into another, leading to duplicate code produced that was not managed by a library. Suppose X set up a library to manage, and share code segments. Then, developers could search for code segments more efficiently. Another advantage of using a library to manage, and share code segments is that the code segment would be upgraded automatically when the library version is upgraded. Upgrading multiple instances of a code segment through a library is typically more time-efficient, than finding every instance of the code segment and replacing it.

An anecdote of costly repercussions from duplicate code being developed was observed between Team Cat and Team Mouse. Team Mouse used Team Cat's project inside of Team Mouse's project. Team Mouse wanted Team Cat to create modules for Team Mouse to use inside of their project. Team Cat refused to create the modules because creating the modules would take up too much of Team Cat's resources. Team Mouse needed the modules and initiated development on the modules themselves, and dedicated months of senior developer time to create them. Team Cat discovered Team Mouse creating the modules, and decided to create the modules themselves without informing Team Mouse. Team Mouse was surprised after the modules were completed because Team Cat had already finished its version. Team Mouse would have ceased to create the necessary modules if Team Cat had made them aware that Team Cat had decided to create them. Team Mouse wasted its time and Team Cat's time because it would have been faster if they had done the modules together. X ultimately suffered financially by paying for duplicate work. This is a prime example of X's failure in teamwork and to demand proper documentation for project status reports to recognize when two teams worked on the same task.

Lack of Hardware Impacts on Development

I observed that X did not have enough newer hardware units to test new software, but X had enough older hardware units to start development of new software. I observed that X's lack of newer hardware units needed to sufficiently test new software caused delays in development for X's projects. Rather than invest in a sufficient quantity of newer hardware units because of its high expense, X created several mock services to mimic the responsibilities the newer hardware units were to fulfill. As mock services were an imperfect substitute for the newer hardware units, defects were expected to be found when tests were performed on the newer hardware units. For example, software integration defects with the newer hardware units cannot be found until the software is tested on the newer hardware units.

Newer hardware units dedicated to test software were often not available at X until later in the development life cycle of a project. This was because the newer hardware units X had purchased that were reserved for projects in later stages of the development life cycle. These projects were prioritized to use the newer hardware units because of their deadlines, and X wanted to avoid expensive missed deadline penalties.

X would offer premature releases of its products to customers, which may have had defects unnoticed by developers because of the lack of tests run on the newer hardware units. I observed that developers at X often had to request access to X's customers' newer hardware units, to investigate defects because of limited availability of newer hardware units. I believe that these requests led customers to lose confidence in X's products.

On the upside, X owning and maintaining older hardware units worked well for projects involved with customers who used older hardware units. On the downside, X's not owning and maintaining newer hardware units made it difficult for X to produce software compatible with newer hardware. X's customers purchased newer hardware and were upset that the newer software was not compatible with newer hardware. X was slow to invest money to buy newer hardware units despite X's employees requests for newer hardware units. X's requisition process for new hardware was slow and difficult. X's employees viewed the process as a discouragement.

Company Culture

I observed that X's employees took pride in their craftsmanship and company culture. I believe X's positive company culture allowed employees to enjoy their workplace, foster better relationships with coworkers, and influence a more collaborative and productive work environment. X had an informal structure that encouraged employees to step into leadership roles when appropriate. X encouraged a cross-team collaborative structure that encouraged employees to help each other, even at the cost of personal time. I believe that X's employees stayed at X because X invested heavily into maintaining a positive work culture. For example, I heard stories of employees that worked unpaid overtime and slept at X to maximize the amount of time they could spend in preparation for a big trade show. While unpaid overtime is not a good thing in general, its existence shows the employees' commitment to working at and for X.

Lack of Documentation

X replaced both hardware-centric solutions with software-centric solutions, and software-centric solutions with newer software-centric solutions. X historically had documentation on its hardware-centric solutions with enormous detail, making transitions from hardware-centric solutions to software-centric solutions easy. I observed that X's software-centric solutions lacked documentation, and the documentation found had little detail, making transitions from software-to-software-centric solutions difficult and time-consuming.

5.1.2 Recommendations 2017

In 2017, I submitted to X's management in a written format this case study's initial findings including the following recommendations:

Enhance and Improve Quality Management System

A Quality Management System (QMS) is focused on consistently meeting customer requirements and enhancing customer satisfaction. QMS aligns requirements with a company's business objectives and strategic direction [8]. International Organization for Standardization (ISO) is internationally known for standards they develop and maintain to ensure the quality, safety, and efficiency of products, services, and systems. ISO9001 is a certification by ISO for QMS to indicate a company has a documented and consistent process. Customers want to work with an ISO9001 certified company, as the certification indicates the company is low risk.

I recommended that X consider initiating ISO9001 certification in order to make X more attractive to customers. The cost of the overhead in ISO9001 could be covered if X received more quantity and lucrative contracts for having ISO9001. For example, a person with more certifications typically charges more than someone with less.

***Improve Status Reporting of Projects to* Managers**

Managers need to determine what actions to take in order to ensure a project's success. It is in the best interest of managers to be aware of a project's current state, in order to be able to take action as soon as possible when necessary.

X's developers constantly dealt with urgent problems, which made it difficult to manage their time between proper documentation and writing code. X's developers focused on writing code and left less time to create accurate project status reports. X's managers often did not have all of a project's necessary information to make beneficial decisions on how to proceed.

I recommended that X define a better process to improve communication of a project's status to managers.

Research Before Actions

It is commonly recognized that more preparation usually results in a better outcome. It is essential to initially investigate multiple potential solutions for the original premise, before a commitment is made to a single solution. Far too often, X would start to write code for a task before detailed research was conducted.

I recommended that X perform dedicated research tasks, such as a Spike task before code is written.

Reduce Training Time for Co-op Students

All new employees need to be trained in X's processes; however, co-op students are a temporary workforce and the cost to frequently train a new co-op student is expensive because of high turnover rates. A formal onboarding process for co-op students would provide more consistent onboarding results and incremental process improvements. A formal

onboarding process could lead to a more consistent and higher quality output from a co-op student and reduce the time to train a co-op student.

I recommended that X spend more time defining and documenting a formal onboarding process for new employees. A new full-time employee also needs to be onboarded, but their onboarding is different from what is needed for a co-op student.

Using CMMI
CMMI is geared towards a solution that will ensure a project stays on time and budget. CMMI will provide a company with a path to the next level of project process management, implementation of clear outlines, and strategic plans to achieve company goals.

I recommended that X adopt a maturity model such as CMMI. X was clearly in a state of chaos, CMMI is known in the SE community, and any change would help allow X to move forward and experience a different approach. CMMI would provide a structure for X to follow to find areas to improve, similar to the way I was able to make recommendations.

5.2 Investigation in 2019

Section 5.2.1 and Section 5.2.2 look at my observations and recommendations from 2017 to how X progressed by 2019.

I made recommendations to X that focused on X's work practice improvements in 2019. Section 5.2.3 discusses each recommendation given in 2019 under the following subheadings: Managing Duplicate Code; Well-Defined Requirements; Focus Hiring Efforts; Understanding Process; Retrospectives; Limited by Bottlenecks; Deadline Planning; Perform Requirements Analysis Sessions; More Productive Meetings; and Discrepancy between PM and Developers.

Management Changes
By 2019, many management changes had occurred that affected X, such as a new CEO at X's parent company in the United States and a new head of X in Waterloo, Ontario. The new head of X in Waterloo also became the new head of X's sister branch based in Toronto, Ontario. The new CEO spent time on site at X every quarter. X increased its communication with X's parent company and X's sister branches, but X continued to function as a lone entity. In 2019, X did not work on the same projects as its sister branch in Toronto but shared resources. For example, X sent Participant M to the Toronto branch twice a month to help with problem solving.

Management with a more inclusive attitude makes a substantial difference. X's new head attempted to improve the relationships between X and X's sister branches. The new head was a long-standing X employee with a good relationship with X's employees, and took the time to learn and understand X's issues. By 2019, the new head of X made a lot of positive changes.

5.2.1 Looking at Observations in 2017 and How It Changed by 2019

Frequent Customer Requirement Changes

In 2019, salespeople and PMs actively tried to minimize the quantity and maximize the quality of changes the customers could make, with some success. Needlessly frivolous and expensive customer requirement changes were no longer accepted by X after a contract was signed. In 2017, X's customers did not follow a strict process to get their desired changes approved. In 2019, customers had to follow a specific process before a customer requirement change request would be considered. X's customer requirement change request process included negotiations for adjustments to timelines and costs; however, in 2019, it should be mentioned that some customer requirement change requests were still being accepted at X's expense to maintain good customer relations.

Insufficient Requirements Given to Developers

In 2019, X started to formalize a Cost-Benefit Analysis (CBA) process because of the confusion caused by insufficient requirements given to developers. X's CBA process ensured that salespeople more carefully considered and documented customer projects before a commitment to a contract could be made. X's employees mentioned that the CBA process resulted in more detailed customer requirements. X's developers interpreted customer requirements more accurately and, in turn, received fewer customer requirement change requests.

Contract Penalties

In 2019, X still dealt with contracts with expensive deadline penalties as mentioned in Section 5.1.1 because they were agreed to previously. For example, one contract's delivery date was in 2019 with a penalty of CAD 80,000 a day for each day project delivery was late. Fortunately, X's developers completed the necessary work in time for the contract, and no known penalties were incurred. X started to ensure that newer contract requirements were more well-defined and limited in scope. X did not cancel contracts when customers asked for requirement changes, but would renegotiate contract terms before a requirement change was accepted.

Customer Requirements Take Priority and Cause Delays

In 2017, a new customer requirements change would take priority over scheduled developer tasks and cause delays to project deadlines. When X started development on Project Moose in 2018, X also managed customer requirements with a new process. Project Moose was a replacement for many of X's older projects that were still in development at the time. Project Moose was designed in a way to be fully customizable on-demand by its customers. Participant M designed Project Moose to mitigate the cost and time to provide a customer with a customized version of its products. Project Moose was thought to be a revolutionary idea by X's stakeholders.

X improved its customer requirements change control by tracking resource over-commitments, better project prioritization, and better backlog management. X's new customer change control process was still being refined and indoctrinated into X's culture by 2019.

Duplicate Code
In 2019, X's developers still did not have a cloud repository to share reusable components across X's teams effectively. X's UX team created and maintained a small-scale NPM library to share SVG images. This NPM library helped the UX team prevent duplicate icon creation, allowed all of X's employees to have the same version of an icon, and gave knowledge of what icons had already been created. I observed that X's developers did not create an NPM library to share code. The copy and paste practice from 2017 continued a practice that could result in duplicate code and was time-consuming.

Project Moose was designed to encompass many of X's older projects by transforming the older projects into large-scale reusable components. In 2019, Project Moose was still years from its anticipated release date.

Lack of Hardware Impacts Development
By 2019, X's budget for hardware purchases became more predictable, and it became less complicated to submit a hardware purchase request. A specific budget was made at the beginning of each fiscal year for hardware. X's employees were encouraged to spend their entire budget on whatever they required, unlike in the past when requests would often be rejected.

Company Culture
X's company's culture was already a point of pride for X. By 2019, X had increased its budget for fun events. As employees interacted more in a relaxed environment, I observed that they tended to build stronger connections and were more open to collaboration on projects, like the adage "The family that plays together stays together".

Lack of Documentation
Detailed internal technical documentation remained a challenge for X from 2017 to 2019, although improvements were made. For example, by 2019, X had dedicated employees to write release documentation on its software-centric solutions; however, the documentation was not completed until the project was ready to be released.

Without up-to-date detailed documentation, X ran a risk that knowledge could be lost. Knowledge can be lost if forgotten, or knowledge that resides only with one developer can be lost if the developer leaves the company. Lost knowledge can lead to enormous negative financial repercussions if knowledge needs to be recreated.

5.2.2 Looking at Recommendations in 2017 as of 2019

Section 5.2.2 looks at how X fared since 2017 regarding my recommendations I presented to X in 2017. X did not follow my 2017 recommendations directly as a result of this case study, however, many positive changes aligned with the recommendations were observed in 2019.

Enhance and Improve Quality Management System

X's Team Indigo pursued ISO9001 certification, as one of X's customers required it as a condition of continuing to do business with X. I believe that Team Indigo benefited from ISO9001 certification because ISO9001 certification forces a company's process to be written down, reviewed, and monitored. Before ISO9001 certification, Team Indigo had an imperfect development process, filled with many manual test steps, and induced risk from its inherently long iteration times. A shortfall of ISO9001 certification is that the certification does not verify the quality of following a process in two ways [34]:

1. Following a process does not guarantee that the follower will achieve the desired results.
2. Certification does not guarantee that the certified company continues to follow a process in the future.

***Improve Status Report of Project to* Managers**

X implemented better processes by 2019, ensuring better quality status reports were presented to managers daily. Managers were in daily meetings with customers to communicate a project's status. Many employees viewed the majority of these meetings as too long. For example, meetings became fixated on small isolated details, which caused too many employees to be held up over tiny details that could be worked out at another time with fewer employees involved. Projects' statuses were more often reported to managers but managers would ask for extensive details in the report that developers felt managers did not need to know during status report meetings.

Research Before Actions

The deadline for one of X's primary customer projects, Project Nano, was set for June 2020. X's networking team developed Project Nano's network. The networking team did not collaborate with other teams at X that worked on Project Nano when they designed its network. By 2019, Project Nano's network design started to reveal defects. Since so much time and investment was already put into Project Nano, X decided not to change course when network inadequacies arose. Due to the inadequacies that arose, many of X's employees were unsure if Project Nano would ever be finished.

In hindsight, more employees outside the networking team could have been involved in Project Nano's network design, as the network affects many other teams. The networking team could have spent more time dedicated to research before writing code and communicating with other teams for more potential solutions. Research could also include the creation of a prototype in order to validate a potential solution.

Reduce Training Time for Co-op Students

By 2019, X's co-op student training programs had improved. I observed that X's co-op students created a program to document and share various information received from

different co-op mentors. This program allowed all co-op students access to all co-op mentor information.

By 2019, X improved its onboarding for co-op students by instituting a dedicated week-long training session to teach co-op students the tools, and languages necessary for their job. X allocated more employees to be co-op mentors. X instituted a feedback process to elicit information from its co-op students on how well their training process went and how to improve it. X was heavily invested in its co-op student program despite the fact that a co-op student's contract is typically four months long. X hoped that one month of training would be sufficient. It would be an incredible achievement for X if they could train co-op students to be productive within one month. To save money, X often hired co-op students with little experience; however, co-op students with little experience are unlikely to be productive after one month of training.

By 2019, X had increased its average number of co-op students hired from four to twelve a term, further emphasizing the need to make co-op training more efficient.

Using CMMI
It is difficult for a company to move up CMMI levels quickly. Being a top-tier CMMI level company requires complete documentation for every line of code written, and every line of code changed, and an enormous amount of money spent annually on one piece of software with an environment that has been essentially fixed for its entire life. High CMMI level companies could achieve low defect rates, but maintaining high CMMI levels is tremendously expensive.

Since X is a small company, I believe it could not afford to invest heavily into using CMMI initially. I believe using CMMI takes a tremendous amount of time, but if CMMI is approached with small continuous improvement iterations towards higher CMMI levels, commitment and involvement would be maintained.

In 2017, I had recommended that X move towards using CMMI to invoke change. In 2019, I believed X was better prepared to start using CMMI; see Section 4.3.2 for more details. By 2019, X started to move towards using Agile methodologies which showed positive results. I recommend X continue to try different ways of improvement.

5.2.3 New Recommendations Made in 2019

Managing Duplicate Code
By 2019, X still did not have a cloud-based library for X's employees to access reusable components easily. Access to reusable components would save employees development time. In addition, X-wide access to a reusable component library could also encourage collaboration, such as employees working together to maintain the reusable components, like the adage "Many hands make light work."

NPM allows developers an alternative to copying and pasting code segments into different codebases. NPM has version control and allows for automated updates. Reusable components can be injected into multiple code bases for use through NPM, helping to avoid excessive manual work.

An example of a library designed to help developers create reusable components shared amongst many projects is Storybook [36]. Storybook allows a developer to create documentation on a reusable component, and provides an interface to test and search for existing reusable components. Storybook allows for the reusable components that are created to be independently injected into codebases through NPM.

Another tool that aids in code sharing is called Bit, found at www.bit.dev. Bit provides a user interface that is easily searchable, and allows shareable components to be independently injected into codebases through NPM.

I recommended that X inject reusable components into multiple projects with NPM. I further recommended that X develop reusable components using a library, such as Bit and Storybook.

Well-Defined Requirements

Co-op students who had mentors assigned to them scored the highest RDMMM level. I observed that co-op students received the most well-defined requirements. I observed that co-op mentors rewrote requirements before passing them to their co-op students to clarify the requirements.

I recommended that X institute a process to ensure well-defined requirements are written. For example, X could institute minimum-information standards for a Jira task that must be filled out before a Jira task can be saved. This Jira task information could include a project's name, objective, acceptance criteria, and priority level.

Focus Hiring Efforts

Full-time employees gave X higher quality and more consistent results than subcontractors, as explained in Section 4.3.1 and Section 4.3.2. For example, Participant M, a full-time employee, created Project Moose, a revolutionary project idea that resulted in a substantial increase in business for X. Project Moose attracted new customers to X, and current customers signed new contracts to purchase Project Moose.

X continued to hire contract workers but improved its interview process, which resulted in higher quality contract workers being hired. Contract employees can quickly scale up a company during peak times and can be let go when they are no longer needed. Despite X's more rigorous interview process, contract workers hired at X still came at a high cost relative to the quality of work they produced.

X had a fixed budget with which to hire co-op students. X's preference was to hire the least experienced co-op students because they cost substantially less than full-time employees. X's preference was to hire more inexperienced, inexpensive co-op students rather than hire fewer co-op students that were more experienced but more expensive. Canada's federal government had a program that provided X with a financial reimbursement per co-op student hired; this reimbursement encouraged X to continue to hire the more inexperienced, inexpensive co-op students. I observed that X would hire experienced co-op students occasionally despite their higher cost if they had worked at X previously and performed well.

UW supplied pay charts to employers on what UW believed employers should pay co-op students, based on a co-op student's experience level and faculty. In 2016, before this case study, X gave returning co-op students a bonus that was 20% higher than what the UW pay chart recommended. By 2017, X had removed the returning co-op student bonus pay to equalize co-op students' pay despite a co-op student's faculty. After X equalized its co-op students' pay, co-op students from different faculties were more motivated to work at X, which resulted in a broader variety of co-op students hired by X.

As an incentive to convince co-op students that previously worked at X to return, X gave them priority in their contract renewal and did not require them to repeat X's co-op interview process.

I recommended that X focus its efforts to hire a higher percentage of quality workers than quantity. I also recommended that X should reinstate its previous co-op student bonus pay program as an incentive for good co-op students to return because, in my opinion, it is less risky to hire a known good co-op student for more money, than to hire an unknown co-op student for less money.

Understanding More of X's RDM Process and X's Projects

X's QA team debugged projects and was a part of X's entire RDM process. X's long-standing QA employees became familiar with X's RDM process and all of X's projects, which allowed a QA member to become a subject matter expert. Subject matter experts were beneficial to X because when X's customers experienced a problem, only one subject matter expert was needed to fix it. For example, when a problem occurs at a customer site, X did not need to send multiple employees; X could send just one subject matter expert to a customer site, which resulted in cost savings for X.

In 2019, I observed that the overall time to complete Jira tasks had decreased. This decrease could be related to two changes X made around Jira task creation:

1. Jira tasks were broken down into smaller work units.
2. Developers at X gathered requirements earlier in a project's development life cycle stages.

I recommended that X encourage all employees to learn about X's entire RDM process and all of X's projects to understand X's end-to-end products better. More employees involved in earlier stages of a project's development life cycle allow more opportunities to spot problems earlier.

I recommended that X require all employees to participate in its RDM process. I also recommended that X teach all employees how they can contribute to its RDM process. For example, developers should be a part of requirements creation to help developers interpret the requirements into code also to help requirements analysts to know where the requirements specification is complete enough.

Retrospectives

Managers try to focus developer time on coding because developer time is an expensive resource. Developers often have little dedicated time to communicate amongst themselves and other stakeholders. It is essential to allocate time for developers to clarify requirements and reflect.

I recommended that X perform retrospectives to allow developers dedicated time outside of coding for communication with other stakeholders.

Limited by Bottlenecks

I observed that certain areas at X, such as DevOps, experienced bottlenecks because only one or two employees at X were skilled enough to complete certain work. Ideally, X should hire more people in high-demand areas that cause bottlenecks; however, skilled employees are expensive, and X did not want to increase their budget to hire them.

I recommended that X assign remedial tasks to unskilled employees, which would allow more time for skilled employees to work on problems that cause bottlenecks. I also recommended that X train more employees in high-demand areas, which would provide X with more employees skilled enough to step in and help alleviate bottlenecks when they occur. I further recommend X teach unskilled employees new skills.

Deadline Planning

I observed that X often underestimated its time to complete a project to attract and secure customer contracts. Due to contract deadline penalties, as seen in Sections 5.1.1 and 5.1.2, X often could not afford to delay a product's release date. I observed that X delivered unfinished products, which resulted in dissatisfied customers. If X had a later release date, I believe X would have delivered a better quality product, but a release date is a major factor in contract negotiations, and X could lose potential customers with later release dates.

I recommended that X use more realistic project deadlines to build better quality products. I believe X could lose potential contracts initially, but would ultimately earn more money by delivering a better quality product in the long run. More investigation to prove this claim is still needed.

Perform Requirements Analysis Sessions

I observed that X's customers' expectations were often unmet, and their visions were often not adequately represented in a product.

I recommended that X perform more requirement analysis sessions with its customers to refine product feature requests, and derive more detailed technical requirements. These sessions would involve cross-functional team participation, group analysis sessions, and creation of high-level design sessions to understand a customer's requirements on a project's architecture, creation of well-defined documentation of a project's requirements, and discovery of test scenarios, with results from these sessions stored in Jira.

Productive Meetings

Meetings take up time and resources, and with X's tight deadlines, it is essential to use all employees' time with productive meetings. Meetings have value when they are adequately tailored to those who attend the meetings. Experts need to be heard, and important information needs to be shared. It is also important to consider who is essential when meeting attendees are selected.

I recommended that X schedule fewer but more productive meetings. For example, X could organize a meeting with its product stakeholders only when a substantial change to a product was considered. The meetings' objective would be to verify if a software's integrity was maintained.

I recommended that X organize smaller subgroups to discuss a project's finer details. I recommended that X create meeting agendas to organize meeting objectives better. I also recommend preliminary meetings to understand the vision, direction, and relative priorities of an upcoming stakeholders meeting when meeting agendas are not clear.

Discrepancy Between PMs and Developers

I found anecdotal evidence in 2019 of a difference between PMs' perceptions and developers' perceptions about requirements the PMs gave to developers. For example, PMs complained that X's developers were too caught up in requirements that were missing finer details, and developers complained that PMs did not provide sufficient requirement details to interpret requirements into code. PMs should have been more open to feedback developers gave them on requirements.

Developers have to take requirements and translate them into code that a computer can understand. Thus, they need detailed requirements specifications, while a PM needs a general understanding of the requirements. PMs most likely do not understand the level of detail a developer needs to complete their job while a co-op mentor, being a developer, does, which explains why co-op students consistently had a higher perception of X's RDMMM level when given requirements by co-op mentors.

I investigated Jira tasks created by Participant R. One Jira task T1 that I investigated stated only: "Autocomplete dropdown does not show all items." T1 did not explain when the autocomplete dropdown should show all items. A developer needs to clearly understand all states and operations any feature needs to create the feature. T1 requires the developer to either speculate T1's desired functionality or ask for clarification because T1's description is incomplete. I investigated Jira tasks created by co-op mentors, and they had substantially more detail than T1; they included diagrams and options on how to complete the task.

I believe X's PMs did not understand how important finer details are for developers to complete their job successfully and avoid redundant work.

I recommended that PMs elicit feedback, especially from developers on how to make Jira task descriptions better, and PMs should adjust their Jira task creation process based on feedback. I also recommended that developers give constructive feedback on what kind of details they expect for Jira tasks. I further recommend that X's PMs be the developer for a project that they have not worked on to give feedback on the requirements they receive, however, many of X's PMs can not program.

6 Considerations for the Future

The following is a list of suggestions for considerations for future case studies:

1. I presented conclusions and recommendations in a formal written report in 2017 to X's managers, which X's managers neglected to read. As a result it is uncertain how influenced the changes between 2017 and 2019 were by this case study through interactions with X's employees.

   Further research could examine the scenario of what happened if X found value from my case study recommendations and made a direct effort to learn from the CMMI and RDMMM level assessments.

2. X followed many Agile principles. Agile values communication, adapting to change, and producing working results. A small business could start to use Agile software development methods. Some may have the opinion that Agile is in tension with CMMI. I believe CMMI activities can be mapped to Agile techniques. For example, CMMI Level 2 requirements management focuses on managing requirements of the project's products and product components to ensure they align with the requirements and project plans. Agile supports several strategies to prioritize requirements such as backlogs and addressing changing stakeholder needs [41].

   Further research could include a case study that examines a small company that uses Agile method to determine its downstream effect on its developers' performance. Future research could also include a case study that takes a company's CMMI level before and after the company starts to use the Agile method.

3. I observed at X that increased use of Jira resulted in better communications between stakeholders, more detailed requirements, and higher developer morale. X had previously used different issue trackers before they used Jira without much success.

   Further research could examine downstream developer performance as a function of the issue tracker a company uses.

4. X's developer performance improvements seen over time could have resulted from developers' increased experience rather than a higher CMMI level or a higher RDMMM level.

   Further research could closely examine how a developer's experience impacts their company's CMMI and RDMMM levels.

5. I observed a positive correlation between project success and the involvement of an experienced, skilled, high job-title employee on a project.

   Further research could look more closely at the difference between a project's success and the involvement of an experienced, skilled, high ranked employee on a project.

6. I would recommend performing this case study again, this time using additional performance processes or metrics to determine how successful a project is such as: writing stable code, writing testable code, reducing code churn, writing simple code, and sharing knowledge. These new performance metrics focus on the project's longevity and consider that developers do more than just programming.

   Further research could involve doing a case study similar to this case study with X but change how developer performance is measured [35].

The following are definitions and examples of different performance metrics:

Writing Stable Code
Writing stable code means to reduce harmful changes that could affect a project and other connected projects. For example, stable code minimizes the downtime on the product for new releases.

Writing Testable Code
Writing testable code means to write code that is both easy to automate and easy to write. Tests act like documentation to quickly show errors to a developer. For example, writing testable code could split code into modules that can be tested individually.

Code Churn
Code churn is how often code changes over time. For example, a code churn objective could attempt to not allow more than 10% of a project's total lines of code to change within two days.

Writing Simple Code
Writing simple code makes a project's codebase easier to test and change. Simple code also helps new team members as it will be quick and easy to learn the project's codebase. For example, simple code could be code that all team members understand.

Sharing Knowledge
Sharing knowledge is essential for team members to learn from each other's mistakes and strengthen professional ties. For example, sharing knowledge could require each employee to share what they learned from a publication with their team once a month.

7 Conclusion

From 2017 to 2019, I found evidence that X's RDMMM level increased. I found substantial evidence that X's CMMI Level increased from 1 to 2. I found substantial evidence that X's developer performance metrics increased.

In 2017, I observed that X's employees had a poor perception of X's performance even when X's parent company praised it for its success. Some employees at X were unsatisfied with X's RDM process and searched for ways to improve it. I observed that X's developers could still complete tasks successfully despite a flawed RDM process; however, it was time-consuming and frustrating for X's developers. I believe X's employees' efforts to improve X's RDM process led X to initiate changes.

In 2019, I observed changes at X that resulted in many benefits: detailed documentation, clearer contract requirements, fewer customer change requests, higher customer satisfaction, increased intergroup coordination, better co-op student onboarding process, increased hardware budgets, and more.

I found CMM's KPAs were an excellent structure to determine which areas to start my observations.

This case study reveals supporting evidence that a higher CMMI level and a higher RDMMM level could positively affect developer performance. However, X's improvements could have been merely coincidental as X did not completely follow my recommendations. So all I can currently conclude is that a chaotic work environment is bad for developers' downstream performance.

9 References

[1] A. Rahma, S. Sahibuddin, S. Ibrahim, 'A Study of Process Improvement Best Practices.' In 5th International Conference on IT & Multimedia at UNITEN (ICIMU 2011) Malaysia, 2011.

[2] V. Faria de Souza, A. L'Erario, J. A. Fabri, E. Genvigir, 'Improvement of Software Process Small Business SPEM: A Case Study on Requirements Engineering' Cornélio Procópio, Brasil.

[3] G. Finzer and G. Finzer, 'How to reduce Coding Defects- Defect Reduction Techniques| Sogeti Labs', *SogetiLabs*, 2017. [Online]. Available: http://labs.sogeti.com/how-many-defects-are-too-many/. [Accessed: 06- Aug- 2017].

[4] Wohlin C, Runeson P, Ho¨st M, Ohlsson MC, Regnell B, Wessel´n A (2000) Experimentation in software engineering: an introduction. Kluwer Academic Publishers, Norwell, MA

[5] 'Software Engineering Institute', *Sei.cmu.edu*, 2017. [Online]. Available: http://www.sei.cmu.edu/. [Accessed: 08- Aug- 2017].

[6] S. Masters, 'Using Organizational Business Objectives to Guide a Process Improvement Program' Software Engineering Institute Carnegie Mellon University Pittsburgh, PA, March 2011 [Online] Available: http://cmmiinstitute.com/sites/default/files/resource_asset/2522_Masters.pdf

[7] Software Engineering Institute – SEI – Carnegie Mellon University. CMMI for Development: Technical report (CMMI–DEV) CMU/SEI– 2006–TR–008 Version 1.2. Pittsburgh, PA, USA, 2006.

[8] International Organization for Standardization - Systems and software engineering: system life cycle processes Ingénierie: ISO/IEC 15288 IEEE Std 15288: 2008. Genève: ISO/IEC, 2008.

[9] SCAMPI upgrade Team 'Standard CMMI Appraisal Method for Process Improvement (SCAMPI) A, Version 1.3: Method Definition Document' Software Engineering Process Management, March 2011. [Online] http://www.sei.cmu.edu/reports/11hb001.pdf

[10] Araújo, E. E. R., Meira, S. R. L. Inserção Competitiva do Brasil no Mercado Internacional de Software 2005. [Online] http://www.softex.br/mpsbr/_artigos/artigo.asp?id=381

[11] A. Shontell, 'How 3 First-Time Founders Turned One-Third Of Their Employees Into Millionaires — And No One Found Out About It', *Business Insider*, 2017. [Online]. Available: http://www.businessinsider.com/googles-120-million-divide-acquisition-2014-10. [Accessed: 08- Aug- 2017].

[12] L. Romão, M. Borba, C. Marquioni, A. Souza, 'Small Business CMMI Implantation: The Importance of Organizational Strategy and Engineering Case Study.' University of the Region of Joinville, Brasil.

[13] S. Schach, 'Software Engineering', by CRC Press Oct. 1990

[14] *Support.microsoft.com*, 2017. [Online]. Available: https://support.microsoft.com/en-ca/allproducts. [Accessed: 08- Aug- 2017].

[15] N. Caplan-Bricker, 'If You Want to Talk Like a Silicon Valley CEO, Learn This Phrase', the *New Republic*, 2017. [Online]. Available: https://newrepublic.com/article/115349/dogfooding-tech-slang-working-out-glitches. [Accessed: 08- Aug- 2017].

[16] S. Garcia, S. Cepeda, M. Staley, G. Miluk, 'Lessons Learned from Adopting CMMI for Small Organizations' U.S Army Aviation and Missile Research, Development & Engineering Center [Online] http://www.sei.cmu.edu/library/assets/garcia-cepeda.pdf

[17] CMMI Institute LLC, 'What Is Capability Maturity Model Integration (CMMI)®? | CMMI Institute", *Cmmiinstitute.com*, 2017. [Online]. Available:http://cmmiinstitute.com/capability-maturity-model-integration. [Accessed: 08- Aug- 2017].

[18] C. Lahoz, J. Camargo Jr. 'A study on elicitation activity of requirements in Projects of the Space Area' São José dos Campos, Brazil

[19] K. Ellis, D. Berry, 'Quantifying the impact of requirements definition and management process maturity on project outcome in large business application development' Springer-Verlag London Limited. March 2011

[20] D. Galin, M. Avrahami, 'Are CMM Program Investments Beneficial? Analyzing Past studies', IEEE software November/December 2006

[21] J. Herbsleb, D. Goldenson, 'A Systematic Survey of CMM Experience and Results.' Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. 1996

[22] S. Bleistein, K. Cox, J. Verner, K. Phalp, 'B-SCP: a requirements analysis framework for validating strategic alignment of organizational IT based on strategy, context, and process.' Empirical Software Engineering Research Program, Australia.

[23] Software Engineering Process Management, 'Appraisal Requirements for CMMI Version 1.3 (ARC, V1.3). Carnegie Mellon, 2011

[24] D. Gibson, D. Goldenson, K. Kost, 'Performance Results of CMMI – Based Process Improvement.' Technical ReportCMU/SEI-2006-TR-004, ESC-TR-2006-04

[25] T. Varkoi, T. Makinen, H. Jaakkola. 'Process Improvement Priorities in Small Software Companies' Tampere University of Technology Information Technology, Finland

[26] Manta.com. (2017) [online] Available at: https://www.manta.com/world/North+America/Canada/Ontario/computer_software--G2/ [Accessed 8 Aug. 2017]

[27] J.D. Herbsleb, D. Zubrow, D. Goldenson, M. Paulk, 'Software Quality and the Capability Maturity Model' Communication of the ACM, January 1997

[28] D. Galin, M. Avrahami, 'Are CMM Program Investments Beneficial? Analyzing Past Studies', IEEE 2006 0740-7459/06/20.00 November/December IEE Software

[29] S. Blanchette Jr., K.L. Keeler, 'Self Assessment and the CMMI-AM - A Guide for Government Program Managers' Carnegie Mellon Software Engineering Institute CMU/SEI-2005-TN-004 [Online] https://apps.dtic.mil/sti/pdfs/ADA441819.pdf

[30] M. Dwyer, 'Overview of the Capability Maturity Model' Kansas State University Course 748 [Online] http://people.cs.ksu.edu/~dwyer/courses/748/resources/cmm-tr25/tr25_o2.html

[31] M. Paulk, B. Curtis, M.B. Chrissis, C. Weber, 'Capability Maturity Model for Software, Version 1.1' CMU/SeI-93-TR-024, February 1993
[32] Atlassian Community, 'Jira Software Alternatives', 2018, [Online] https://www.atlassian.com/software/jira/compariso
[33] Members of the Assessment Method Integrated Team, 'Standard CMMI Appraisal Method for Process Improvement (SCAMPI), Version 1.1: Method Definition Document', Handbook CMU/SEI-2001-HB-001, December 2001
[34] British Assessment, 'What are the requirements for ISO 9001?', Company No. 11135335, The British Assessment Bureau, [Online] https://www.british-assessment.co.uk/insights/requirements-for-iso-9001/
[35] T. Osbourn, '6 Essential KPIs for Software Development Teams', January 21, 2020 [Online] https://textexpander.com/blog/kpis-software-development-teams/
[36] Nefarious Planetary Meddling, '@storybook/react', July 20 2021 [Online] https://www.npmjs.com/package/@storybook/react
[37] Tutorialspoint Simply Easy Learning "CMMI Maturity Levels", July 25, 2021 [Online] https://www.tutorialspoint.com/cmmi/cmmi-maturity-levels.html
[38] W. Allison, 'Understanding the SBA's Small Business Definition', December 13th 2020 [Online] https://www.lendingtree.com/business/sba/small-business-definition/
[39] BenchmarkLaw, 'What is the definition of a small business in Canada?', September 28th 2019 [Online] https://www.benchmarklaw.ca/2019/09/28/definition-of-a-small-business-in-canada/
[40] Microsoft, 'Visual Studio Code', September 27th, 2021 [Online] https://code.visualstudio.com/
[41] Mapping CMMI to Disciplined Agile, December 5th 2021 [Online] https://www.pmi.org/disciplined-agile/mapping/cmmi

10 Appendix A

10.1 RDM Questionnaire

1. Who are you? (Name / NickName / Participation number) How long have you been working at the company? In what department(s)? On what product(s)?
2. Using your most recent strategic project, how well do you believe your company accomplishes various tasks involved in a good RDM process? Rate on a 5 point scale

Factors Related to RDM Quality	Very low skill level	Low skill level	Moderate skill level	High skill level	Excep-tionally high skill level	N/A
Describing project goals and objectives in concise, clear, and unambiguous terms						
Facilitating cross-functional group sessions where requirements were discovered						
Conducting efficient meetings and making effective use of stakeholder time						
Accurately documenting the business process behind the application						
Describing the information flow and key data needed by users at any given time in the business process						
Assuring that the scope of the project neither significantly changed nor had major in-scope functions moved to follow-on phases of the project						
Uncovering project interdependencies or issues that need investigation						
Clearly describing project risks and assumptions						
Presenting the results of analysis in clear, well-organized, and readable documentation						
Assessing change requests: determining the impact of these on scope and cost, and the impact of systems changes on business processes						
Achieving consensus on requirements amongst stakeholders						
Getting requirements documented in a short, concentrated period of time						

3. Using your most recent strategic project, how well do you believe your company accomplishes various tasks involved in factors related to the organization? Rate on a 5 point scale

Factors Related to Organization	strongly disagree	disagree	neither agree nor disagree	agree	strongly agree	N/A
Our organization has a formalized approach to doing business requirements which is consistently followed by business analysts on projects						
Our organization has defined standards for business requirements documentation quality and assesses the work of analysts against these standards on projects						
Our organization treats business analysis as a profession and has trained staff dedicated to this function						
Our organization can predict how much stakeholder time will be needed and which stakeholders will be involved in the requirements phase of a project						
Business requirements are traceable, and well integrated into testing at our organization						
Stakeholders feel that the process of gathering/eliciting and documenting requirements is efficient at our organization						
Our organization is excellent at transitioning requirements from business departments into the information technology department						
I believe the automated business analysis tools that we current have in place are excellent at helping us elicit requirements						
I believe the automated business analysis tools that we currently have in place are excellent at helping us manage requirements and requirements change						

4. Please describe who you think is supposed to be giving you project requirements? Who gives you project requirements? (If there is a difference between the two, do you feel like this is helping you be more or less successful, and why?) (Question only added in for 2019)

5. Are there any thoughts you would like to add? From the survey questions or outside of the questions able to answer that you think will help the developer understand requirements?

10.2 Performance Metric Data Table

Category	Error Density	Productivity	Rework	Cycle Time	Schedule Fidelity	Error Detection effectiveness	Project Success
Result							

11 Appendix B

11.1 Script

```
@echo off
setlocal
set /a totalNumLines = 0
for /r %1 %%F in (*.ts *.html *.less *.scss *.vue) do (
  for /f %%N in ('find /v /v '' ^<'%%F'') do set /a totalNumLines+=%%N
)

echo %totalNumLines%

set /a codetests = 0
for /r %1 %%F in (*.spec.ts) do (
  for /f %%N in ('find /v /c '' ^<'%%F'') do set /a codetests+=%%N
)
echo %codetests%
```

12 Glossary

Acceptance criteria are the conditions that need to be satisfied before a task can be considered complete.

Agile is a development framework that focuses on improving communication, adapting to change, and producing minimal viable products.

Backlog is a list of tasks that represents outstanding work in a project.

Backlog grooming is a regular session in which backlog items are discussed, reviewed, and prioritized.

Bottleneck is a single source that limits the overall output, similar to the way the narrow neck of a bottle will slow the flow of liquid.

Bit is an application that helps to build and reuse components. Found at: https://bit.dev/

Business analyst is a person who specializes in conducting research and aids with requirement gathering and process improvement.

Change control is when all requests to change anything from the approved baseline are captured, evaluated, and then approved, rejected, or deferred.

Co-op student is a post-secondary student employed by a company, typically as a part of the student's degree requirements.

Code churn is how often code changes over time.

Company culture is a set of shared values, goals, attitudes, and practices that characterize a company.

Configuration Management Baseline is an agreed upon description of the attributes of a product, at a point in time, which serves as a basis for defining change.

Cost–Benefit Analysis (CBA) is a systematic process to estimate alternatives to determine options that provide the best approach to achieving benefits while minimizing costs.

Customer requirement is a feature or specification that the customer considers essential.

Customer requirement change is work requested by a customer that is outside the scope of the original contract.

Developer is a person who writes, tests, debugs, and maintains computer applications.

Development and Operations (DevOps) is a department specializing in information technology that improves the systems development life cycle and assists in continuous delivery.

Dogfooding is the idea of the developer of software using internally the software it has produced.

Domain is a knowledge area that is bound by a similarity.

Duplicate code is when a developer creates the same functionality multiple times, resulting in higher development costs in code creation, testing, user experience review, gathering requirements, and more.

Fagan-style Inspections is a process of finding defects in documents during different stages of the software development life cycle.

Hero cookie is a reward concept. A hero cookie is given out for recognition and thanks to an employee each time the employee goes to extraordinary efforts to achieve success. There is no monetary value associated with a hero cookie.

Integrated unit test is a test that combines different modules in the application and tests them as a group to see if they are working together as intended.

Jira is an issue tracking software created by Atlassian that allows for project management and bug tracking.

Jira board is a tool that unites teams around a single goal and promotes iterative, incremental delivery.

Jira task is a work item that needs to be completed and recorded in Jira. A Jira task represents a requirement. Jira tasks can have points associated with them representing an estimate on how long the task will take.

Jira roadmap is used for planning large pieces of work that span over months for a single project.

Kubernetes is an open-source program for managing containerized workloads and services.

Likert Scale is a unidimensional scale used to collect participants' attitudes and opinions.

Lunch-and-Learn is a voluntary meeting to share information through either training or presentations. Lunch-and-Learn is not necessarily directly related to work and often occurs during unpaid work hours.

Mock-up is a simplified version of an object meant to simulate the behavior of a real one.

Node Package Manager (NPM) is a package manager for the Node JavaScript platform. It places modules in a specific way such that nodes can find them and allow them to be used by software applications.

Ordinal scale is a way to sort the ranking and ordering of the data without establishing the degree of variation between them.

Pair programming is a software development technique in which two programmers work together at one workstation.

Person–month is a metric for expressing the effort (amount of time) personnel devote to a specific project.

Peer reviews are the act of checking fellow programmers' work for code mistakes.

Product Manager (PM) is a person who oversees one specific project at a time. A PM communicates directly with customers to understand their needs and work out potential solutions at a high level.

Project iteration is a timebox in which work items are fully developed and tested.

Product Line (PL) is a group of related products all marketed under a single brand.

Product Line Manager (PLM) is a PM who oversees multiple projects in a product line.

Quality Assurance (QA) is a team of people who work on improving and maintaining a product's quality.

Quality Management System (QMS) is focused on consistently meeting customer requirements and enhancing satisfaction.

Requirements Definition and Management (RDM) is a synonym to RE, and that except for when this document is quoting or paraphrasing other work, this document is using "RDM".

RDM Maturity Model (RDMMM) is a maturity model specifically designed to develop better RE.

RDMMM level is derived from the median of a specified set of participants' numerical values obtained from a RDM questionnaire.

Requirements creep is when project requirements increase so that they become difficult to achieve within the desired timeframe.

Requirements Engineering (RE) is the process of eliciting, analyzing, defining, documenting, and maintaining requirements within the engineering design space.

Retrospective is a dedicated time for employees to brainstorm ideas for improvements, create shared understandings, and self-reflect. A retrospective can also be used to critically examine a development process to make it more efficient.

Returning co-op student is a co-op student who had worked at a company in the past and was contracted to work at the same company again as a co-op student.

Reusable component is a code segment that can be used multiple times in the same or different projects, particularly with a product line.

Salesperson is a person responsible for selling a company's products to customers.

Senior co-op student is a co-op student that has worked at a company for at least a year. A senior co-op student typically receives requirements directly from PMs, was not assigned a co-op mentor, and was typically assigned tasks that are more challenging than those assigned to co-op students but less challenging than those assigned to full-time employees.

Scalable Vector Graphics (SVG) is a unique type of image format for producing images that can be scaled to any resolution without loss of or gain of resolution.

Scoping is deciding what features or use cases will be in the scoped development.

SCRUM is a process framework to manage product development and help teams work together.

Spike task is a Jira task to dedicate time to answer questions and gather information on malfunctioning code.

Sprint is a timeboxed period during which teams attempt to complete a specified amount of work, e.g a spike task.

Stable code is code that results in minimal interruptions to a project and other connected projects.

Stakeholder is a person connected to a project that can either affect or be affected by a project, such as an investor, employee, customer, or supplier.

Standup is a short daily meeting to check on the team's status.

Story point is an arbitrary value representing a weight decided upon by a team for estimating how long a Jira task will take to complete.

Subject Matter Expert (SME) is a person who is an authority in a particular area or topic.

Team lead is the team member responsible for facilitating the team, obtaining resources for the team, and sorting out the team's problems.

Test code coverage is the percentage of code that is executed when a test is run in a software project.

User Experience (UX) is how users interact and experience a product, system, or service.

Unit testing is the process of creating programmatic tests for a part of a program to ensure that the part is functioning correctly. Unit testing is typically performed by a developer.

Work-to-rule is a form of job action in which no employees does more than the minimum amount of work required.