EVALUATING SOFTWARE PROJECT MANAGEMENT QUALITY
USING A CASE STUDY APPROACH

A graduate project submitted in partial fulfillment of the requirements
For the degree of Master of Science in Software Engineering

By

Paleny Topjian

May 2015
The graduate project of Paleny Topjian is approved:

_________________________________________ ______________
Dr. Robert Lingard                          Date

_________________________________________ ______________
Dr. Taehyung Wang                            Date

_________________________________________ ______________
Dr. George Michael Barnes, Chair             Date

California State University, Northridge
ii
# Table of Contents

Signature Page ........................................ ii
Abstract ............................................ v

Section 1: Introduction ................................. 1
   Objective of Research .............................. 1
   Overview .......................................... 1

Section 2: Proposed Software Company ............. 3
   Overview .......................................... 3
   Software Installation and Accessibility ........ 3
   Business Model ................................ 3
   Software Training and Support ................. 4
   Software Customization ......................... 4
   Data Transfer .................................... 4
   Program Defects and Updates .................. 5
   Management and Structure ....................... 5
   Quality Management ............................. 5

Section 3: Metrics ..................................... 7
   Overview of Metrics ............................. 7
   Using Metrics to Measure the Quality of Software Management .... 7
   Application to Case Study ....................... 8
      How to Develop a Metric .................... 8
      Proposed Metric to Measure Quality of Management in Proposed Company ... 9
   Measure of Success .............................. 10

Section 4: Requirements ............................... 11
   Overview of Requirements Engineering ........ 11
   Requirements in Agile Software Development ... 12
   Application to Case Study ....................... 13
   Evaluation and Proposal of a Requirements Management Program .... 16
   Measure of Success .............................. 17

Section 5: Risk ........................................ 20
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overview of Risk Management</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Application to Case Study</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Evaluation and Proposal of a Risk Management Program</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Measure of Success</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td><strong>Section 6: Planning/Estimation</strong></td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Overview of Time Management</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Implementation of Time Management</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Application to Case Study</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Designing an Agile Development Environment for Time Management</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Evaluation and Proposal of a Time Management Program</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Measure of Success</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td><strong>Section 7: People</strong></td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Overview of People Management</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Application to Case Study</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Evaluation and Proposal of a People Management Program</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>Measure of Success</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>Conclusion</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>41</td>
</tr>
</tbody>
</table>
Abstract

EVALUATING SOFTWARE PROJECT MANAGEMENT QUALITY USING A CASE STUDY APPROACH

By

Paleny Topjian

Master of Science in Software Engineering

Creating quality software within time, budget, and scope constraints is an ambitious undertaking. Since the beginning of the software industry’s history, people have been trying to implement processes to ensure success, and yet there are still many failed software projects. An important factor for the success of a software project is quality management.

This graduate project demonstrates the ability to exercise knowledge of the software development process and software management by creating a “thought experiment.” A hypothetical software company is presented to evaluate the quality of management as a case study. A novel metric is developed based on the GQM paradigm in the form of survey questionnaires. Improvement programs are presented based on the results of the surveys.
Section 1: Introduction

Objective of Research
It is common knowledge in the industry that delivering a software within the iron triangle is a rare and respected accomplishment. The iron triangle refers to the project’s time, cost, and scope. They represent the ideas to which software projects must adhere to in order to create quality software on time and within budget. However, Edward Yourdon argues that “a properly managed project, in a mature software engineering environment, managed by a competent manager, can repeatedly deliver a software system on time, within cost, and satisfactory to the user [1].”

The focus of this graduate project is on evaluating the quality of software project management through a case study. After opening with an overview of Project Management, Section 2 describes a hypothetical software company on which the case study is implemented. Section 3 proposes a metric developed to measure the management quality of the proposed software company. This metric is then implemented in Sections 4 through 7 in four areas of software management. Each area is first introduced, then evaluated using the metric introduced in Section 3, based on its implementation on the proposed software company. Evaluation is done by surveys created based on literature review of the respective subject matter. The surveys are collections of statements that require responses on a 1-10 scale. There is one survey per area of interest covering the topics of requirements engineering management, risk management, planning and time management, and people management. The surveys are answered as part of the “thought experiment” this graduate project proposes. The answers are representative of the type of responses expected from a team member of the proposed company. And finally, the last section summarizes the benefits of the project and the insights gained from this “thought experiment.”

Overview
Project Management is defined by the Project Management Institute (PMI) as, “the application of knowledge, skills, tools, and techniques to project activities to meet the project requirement.” PMI recognizes 42 project management processes grouped into five “Process Groups” identified as: Initiating, Planning, Executing, Monitoring and Controlling, and Closing. Initiating is defined as recognizing when a project needs to begin and committing to its start. Planning is devising and maintaining a workable scheme to accomplish the business need that the project was undertaken to address. Executing involves coordinating people and resources to carry out the plan. Controlling processes is making sure that project objectives are met by monitoring and measuring progress and taking corrective action when necessary. And finally, closing processes formalizes acceptance of the project or phase and brings it to an orderly end [2]. A Guide to the Project Management Body of Knowledge (PMBOK Guide) assigns the following points as the main criteria for managing a project: identifying requirements, addressing the various needs, concerns, and expectations of the stakeholders as the project is planned and carried out, and balancing the competing project constraints including, but not limited to scope, quality, schedule, budget, resources, and risk [2].
The IEEE’s Adoption of this PMI Standard states that “the knowledge and practices described are applicable to most projects most of the time, and that there is widespread consensus about their value and usefulness. It does not mean that the knowledge and practices should be applied uniformly to all projects without considering whether they are appropriate, [3]” a statement which applies directly to this graduate project. The idea is to investigate the following considerations if a case study were to be conducted for a specific type of Software Company to assess the quality and effectiveness of their project management:

- What are the areas of interest that need to be considered?
- How would the existing strategies be evaluated?
- What changes would be proposed?

The baseline for the qualifications of a competent project manager comes from [1], “A competent project manager is someone with the knowledge, skill, and experience to lead a project to completion effectively and efficiently.” The listed responsibilities are endless, but a few notable examples given include managers’ responsibilities [1]:

- To build and maintain positive relationships between customers and stakeholders
- To provide support and lead the development team while managing the software development effort by mitigating risks
- To control staffing and team building so that team members’ full potentials are realized and utilized
- To tailor project frameworks so members believe and support them while simultaneously coming together as a unified team
- To facilitate, resolve conflicts, and communicate for the purpose of defining a common set of project goals and processes

After surveying methods for conducting case studies, and proposing a hypothetical software company to evaluate, the paper uses the above mentioned duties as a reference for defining what areas of management to study, review against the proposed company, and evaluate for improvement.
**Section 2: Proposed Software Company**

**Overview**
The proposed software company that is going to be taken into consideration has the following characteristics. The company is fairly established, having been around for ten plus years. It offers one or more working software products catered to a specific niche industry. It employs around thirty workers consisting of a sales team, management, development (software programmers), IT, and software trainers. At least one of the existing software products offered for purchase is fully functioning with an average of one actual bug claim made by customers per month. The term “actual bug” will be explained in further detail in a later section. The normal cycle that a customer experiences after a purchase is to have the software installed by a technical or IT department personnel. This department can also deal with any hardware installations related to the system such as scanners, printers, etc. They can also help with networking in case the program is to be accessed by multiple users at the same time, such as setting up a host server, VPN’s, remote desktop connections, etc. After being setup, the customer then receives a set amount of training to learn the system in addition to a set amount of software support which allows them to call in for assistance with any questions or issues they have. After some time, they are expected to be self-sufficient and proficient in using the program.

**Software Installation and Accessibility**
The installation follows the purchase of the software which comes with a limited number of user licenses. Normally a technician remotely accesses the customers’ server or host computer that acts as a server to install the program for a single user. If there are any other local stations that will also be using the program, the technician will also connect to these stations and setup connections to the server from each local station, using hardwire network connections or remote desktop connections. Wireless connections are avoided to prevent data corruption in case of a loss of internet connection.

**Business Model**
The basic business model which keeps this type of software company afloat is to collect funds for software and possibly hardware support. Aside from paying a standard premium for the software itself, customers pay to renew their support annually for a year’s worth of unlimited support. If they choose not to pay the yearly support, they can pay on an hourly basis per each phone call. The customer receives a limited time offer from the time of purchase for software training and support. These training sessions are typically offered remotely via remote support access and over the phone. A customer can choose to pay extra to have support team members go on site for training (see section on Software Training and Support), or for technical setup or issues. The software has a default standard which is the basic program with no changes or alterations made to tailor the specific customer. Customers can choose to pay for any modifications to the default program (see section on Software Customization). The company also offers data transfer services at varying rates (see section on Data Transfer) if the customer is coming from a different program and needs their data to live in the new program.
**Software Training and Support**

Software training begins after the software has been purchased and installed. A software support member trains the customer how to use the system. As mentioned earlier, these trainings take place over the phone and by connecting to the customer’s computer remotely to show them the program directly on their monitors. At a higher cost, the customer can request a training on location. The trainings are limited to a pre-set number of hours. Additional training rates may be negotiated depending on how many additional hours of training are requested. Once these training hours are completed, the customer is allowed to call back for minor questions and issues under their software support but not for any additional training for new employees for example. For this, they would need to purchase additional training time.

**Software Customization**

If the customer wants to make changes to the standard default program in order to solve a business need that the current version of the software does not support, they can discuss program customizations. The standard protocol for these customizations includes discussing the changes with a software support member who is responsible for understanding the business need for the change, how the modifications can be incorporated with the existing system, and gathering the requirements. Based on the information they have gathered from the customer, they then get a quote from a programmer on the development team to get an estimated timeframe, meaning how many programmer hours it would take to develop the customization. Logistics such as testing time, release of a new executable, and time it would take to update the customers’ system are taken into consideration when providing a quote. If the customer agrees to the provided quote, they pay for the project to commence. Once the customization is ready, a software support member will update the customer’s system with a new executable and go over the changes with the customer.

**Data Transfer**

Since the company has been around for so long, it is ready to deal with competing programs in the same market. As an incentive to the new customer, data transfer services are offered which have been developed over the years. Different scripts have been written to help migrate data over from different programs into their own software. The data is not always clean, meaning it is not guaranteed that the customer entered data correctly into the old software as the program was meant to be used. This means that a data transfer will not always guarantee that the data will sit correctly in the respective fields. It requires time from the software support team to analyze the data before it is brought over to the new system, and once again after the transfer. If data has to be cleaned or modified on either end, the customer should receive a quote that is appropriate to the amount of person-hours that will be spent working on the data transfer. There might not always be a script available depending on what program the customer is migrating over from, in which case a new script has to be developed by a programmer which inevitably increases the charge of the data transfer service. The new script needs to be tested and the transfer is probably run several times before all the bugs are cleaned and the data is smoothly transitioned. The customer can always choose not to accept the service and enter their data into the system manually.
Program Defects and Updates
As mentioned in the overview, the program is pretty well established and customers find on average about one actual bug per month. By actual bug, it means that there may be more claims made, but they are not actual bugs, they tend to be the customers’ mistake of running a query incorrectly for example, or incorrectly entering their data thus getting unexpected results. In this case, the software support comes into play to find and identify the issue and train the customers how to use the program properly to get the expected outcome. When a customer claims to have a defect in their program, the software support inspects the issue to make sure that it is in fact an actual bug, and if so, relays the issue to a programmer who resolves the issue and applies for the release of a new executable. When the update is released, a software support member makes a backup of the customer’s system and replaces the executable with the update.

Management and Structure
In this proposed type of software company, management does not directly play an active role. Team members are usually left to work autonomously. Team meetings are few and far between which has led to miscommunication when multiple members work on the same customer. Documentation is limited to notes voluntarily entered into the central system by team members debriefing what they worked on with the customers during a particular session, which other members can view for reference.

Quality Management
If the proposed company were to be appraised by quality control measure such as Software Engineering’s Institute Capability Maturity Model (CMMI) [4], its maturity level would probably be categorized as level 1 or 2.

The CMM is a five-level model that attempts to quantify a software organization’s capability to constantly and predictably produce high-quality software products. The model is designed so that capabilities at lower stages provide progressively stronger foundations for higher stages. Each development stage or maturity level distinguishes an organization’s software capability. For each maturity level there are associated key process areas (KPAs). The KPAs identify the requirements for achieving each maturity level [4].

Level 1 represents the beginning stages of a new process defined by chaos, success based on “individual efforts,” and no KPAs. Level 2 shows a little more organization with the implementation of at least “basic project management process to track cost, schedule and functionality,” tools and processes to support repeatability of similar projects, and KPAs which include:

- requirements management
- software project planning
- software project tracking and oversight
- software subcontract management
- software quality assurance
- software configuration management
The goal of assessing the proposed company using the metric proposed in Section 3, is to be able to bring the company up to a Level 3 maturity level. Ideally, this would be accomplished if the proposed company were to implement the improvement plans pitched in the areas covered between Sections 4 through 7. Level 3 would be characterized by defined, documented, standardized, and integrated processes used by both management and team members throughout the company with KPAs which include:

- focused and defined organization process
- training programs
- integrated software management
- software product engineering
- inter-group coordination
- peer reviews

The next section proposes a metric used to assess the areas of software management presented in sections 4 through 7.
Section 3: Metrics

Overview of Metrics
There are many definitions of Software Metrics. Listed here are a few of the important ones that provide insight into the way metrics are used in today’s software industries [5]:

- Software metrics provide measurement for the software and the process of software production, giving quantitative values to the attributes involved in the product or process.
- Software metrics are to give the attributes extracted from the software product, software development process, and the related resources, some quantitative descriptions.
- Software metric is a function, with input as the software data, and output as a value which could decide how the given attribute affects the software.

Ever since the early start of software programming history, people have developed different metrics to measure different aspects of the software development process and product, whether it has been the number of lines of code to measure program size, or programmer hours/month to measure developer productivity rates. These metrics have been developed to acquire a deeper understanding of software development, to help be able to deliver appropriate management of software, and to improve the software design procedure. Measurement is a mechanism for creating a corporate memory and an aid in answering a variety of questions associated with any software development, like how much will a new project cost? Are certain types of errors commonplace? Or what techniques will minimize current problems [6]?

The areas of software which need to be measured are process, project, and product. Process metrics typically used by high level managers to obtain development status include maturity, management, life cycle, product ratio, defect ratio, etc. Project metrics used to improve the quality of the product include scale, cost, workload, status, production power, risk, the degree of satisfaction from clients, etc. Product metrics used to understand and control the quality of the product includes reliability, maintainability, product scale, software complexity, portability, documents, etc. With the ability to measure process, project, and product, we are able to understand each attribute allowing us to select and determine appropriate baselines for evaluation, forecast, control, and improvement [5].

Using Metrics to Measure the Quality of Software Management
Quality software management is vital for the success of software development projects. Metrics is an important tool for managers to help them stay on budget, on time, and within the scope of the project. Without the data that metrics provide, managers would have a hard time estimating and controlling project costs, scheduling project deadlines, or monitoring project requirements, just to name a few. But beyond the metrics created to provide managers with the information they need to make projects successful, there also need to be metrics which measure the quality of the management itself. This is important because no matter how much data a manager is equipped with, if they are not capable of utilizing the information to their advantage, they can drive projects to failure. As the
authors of *Quality Management Metrics for Software Development* [7] point out, “Program-management tools have been developed to assist the program manager in estimating the cost and schedule of software programs. However, the estimation tools available assume consistent and high-quality program management.” One of their examples uses COCOMO as an illustration of this problem. COCOMO is one of the earliest used software project cost-estimation models. It uses different factors to predict the budget of projects, none of which take the quality of project management into account [7]. The entire estimation is based on the assumption that the project has exceptional management, which is a very naive assumption to accept. If the quality of the software program management were measurable and available as input to costing and scheduling tools, the resulting estimates could pinpoint areas of software program management in which improvement needs to be made, thus if two programs scored equally on product and process metrics, the quality of their management could provide a more comprehensive look at the software program [7].

**Application to Case Study**

The following section will consider the development of a new metric that will be used to measure the quality of the software management in the Proposed Software Company. Sections 4 – 7 of the paper are implementations of this metric to survey team members with regard to their assessment of the software management of their company.

**How to Develop a Metric**

There are a number of different approaches to picking, defining, and using metrics based on appropriate models and goals. This paper uses the Goal/Question/Metric (GQM) Paradigm – a mechanism for defining and evaluating a set of operational goals, using measurement [6]. A large variety of software goals, defined from a variety of perspectives include the customer, the project, and the corporation [6]. An example of a management goal could be the need for correct distribution of work between capable team members based on their strengths and talents.

The GQM Paradigm [6] allows an organization to:

1) Develop a set of corporate, division, and project goals for productivity and quality, e.g., customer satisfaction, on-time delivery, improved quality
2) Generate questions (based upon models) that define those goals as completely as possible in a quantifiable way
3) Specify the measures needed to be collected to answer those questions and to track process and product conformance to the goals
4) Develop mechanisms for data collection
5) Collect, validate, and analyze the data in real time to provide feedback to projects for corrective action and analyze the data in a post mortem fashion to assess conformance to the goals and make recommendations for future improvements

Templates and guidelines are offered in the GQM process for setting goals and developing questions and metrics to help provide direction throughout the process. To define a goal, the following template from Victor Basili’s *Software Modeling and Measurement: The Goal/Question/Metric Paradigm* [6] is completed with the appropriate choice of words:

---

8
GQM Template for Defining a Corporate/Division/Project Goal

*Purpose: meant to define the object(s) of study, what we are going to do and why we are doing it*

Analyze some (objects) for the purpose of (why)

*Perspective: meant to define a particular angle or set of angles for evaluation*

with respect to (focus) from the point of view of (who)

*Environment: meant to define the context of the study by defining all aspects of the project so it can be categorized correctly and the appropriate set of similar projects found as a basis of comparison*

in the following context (problem factors, people factors, resource factors, process factors).

Proposed Metric to Measure Quality of Management in Proposed Company

Since this paper focuses on the evaluation of project management, the metric used throughout the paper, is a collection of original surveys created to assess management, from this point on referred to as Targeted Management Improvement Metrics (TMIM). Following the GQM procedure, Sections 4 – 7 will first develop a set of management goals for productivity and quality, then generate questions that define those goals in a quantifiable way. The measures needed to track process and product conformance to the goals will be identified in the overview portion of the respective section. The metric developed and utilized in this paper for data collection is based on the Quality Management Metric (QMM) presented in [7], which evaluates management quality based on four areas believed to be the most important: requirements management, people management, risk management, and planning/estimation management. One of the main differences is that the QMM “does not provide specific feedback guidance for the program managers,” because its results show high level feedback within the four areas mentioned. The metric developed in this paper - TMIM - also assesses managers based on the same four criteria, requirements management, people management, risk management, and planning/estimation management. But the questionnaires are broken up into more specific sub-sections within each of these four areas. The results can be used to provide targeted feedback and explicit areas of improvement as the GQM paradigm requires.

The QMM is computed from the quantitative answers to a structured set of inquiries developed from searching literature, interviewing senior program managers, and conducting focus group meetings to come up with a questionnaire consisting of two parts:
(1) a set of paired choices between statements that reflected possible management actions on a software program, and (2) a set of questions requiring a yes, no, or not applicable answer [7]. For the set of paired choices, the person filling out the questionnaire has to choose one of the two statements that best describes their program, an example of which was, “Estimates by algorithmic methods/Estimates by analogy.” The yes-no-n/a questionnaire standardizes responses for easy comparison by assigning a point value based on the relative importance of the question. An example here was, “PM attempts to motivate individuals on the program team.” The QMM survey was administered to selected program managers and software developers, whose raw QMM scores were weighted, converted to a 1-10 scale, and then compared to subjective success scores estimated for the same programs by the same survey subjects. The point totals from each of the two questionnaire parts per section were entered on the QMM Summary Score Sheet. Point totals for part one and part two were then added together to determine the total points for each section. These were multiplied by their relative Importance Coefficient (IC) – determined by focus groups and interviews of software professionals – to yield a weighted score. After weighted scores were determined for each of the four sections, they were summed together to yield the QMM score [7].

Unlike QMM, TMIM does not concern itself with trying to aggregate a total score. TMIM is a set of statements organized into specific sections within the management areas of requirements, people, risk, and planning/estimation, which elicit responses on a scale between 1 and 10, 1 corresponding to the answer “Not at All,” 5 corresponding to “Neutral,” and 10 corresponding with “Strongly Agree.” The statements are generated based on important process milestones as gathered from literature review. Responses to each statement are the basis for a process improvement suggestion in the subsequent management improvement program.

As in the final step of the GQM paradigm, feedback for corrective action, assessment to conformance to the goals, and recommendations for future improvements are provided in Sections 4 – 7, based off the example data collected from these TMIM.

**Measure of Success**

The idea of creating this metric and responding to the questionnaires is to be able to demonstrate clear understanding of the software development process and software management. Each section from this point on is an in-depth evaluation of four topics of software management using TMIM. These topics are deliberately chosen as areas believed to be important for the realization of a sustainable and successful business model for a specific type of software company as the one described previously in the section titled “Proposed Software Company.”
Section 4: Requirements

Overview of Requirements Engineering
Requirements Engineering is the process of gathering the specifications and needs of the users or customers in a way that clarifies and brings order to these needs so they may be translated correctly into the product being built by the developers. According to The Role of Requirement Engineering in Software Development Life Cycle [8], “Requirements Engineering (RE) is the most important area of Software engineering and possibly of the entire software life cycle.” The requirements are the foundation upon which the software is built, therefore if the foundation is not solid, the software is going to have a hard time being sturdy as well. If the requirements are not gathered correctly, or if they are constantly changing, it becomes a challenge to develop a product that executes the desired functions and produces the needed outcomes. These collected requirements may act as the binding agreement between the developers and customers as to what is expected from the resulting produced software. It is therefore important that the business needs or desired goals which sparked the need for a solution in the first place are well understood and will be resolved. A great deal of research regarding requirements engineering suggests that errors found in requirements are the most costly errors [8], more so than other stages of the development process such as coding errors or testing errors, which adds to the importance of complete and correct requirements elicitation and documentation.

There are a number of existing methodologies for requirements analysis such as abstracting and portioning approaches and domain-based approaches. Abstracting and portioning focus on creating models, which are useful for the developers. The domain-based approach concentrates on understanding the system problem domain and defining the boundary of the system, which is better tailored to capture the users’ needs and wishes [8].

For the purposes of this paper, it makes more sense to take the domain-based approach. Creating Object Oriented system models are useful for developers but not for management. Management cares more about the correct identification and solution of the customer’s problem scope. To use this approach The Role of Requirement Engineering in Software Development Life Cycle [8] shows different diagrams that are useful for clarifying the requirements. The Domain Components Diagram can be utilized to help brainstorm identifying different viewpoints of the system. This diagram includes the different stakeholders that will be using the program, or have some role in the business using the program, as well as the different services that need to be considered as part of the system. The Problem Scope Diagram can help identify the problem scope of the system by illustrating the main objective of the system, the inputs feeding into the system, and the required outputs produced by the system. The Event Trace Diagram goes through the workflow between the stakeholders. Once the different viewpoints, the services each viewpoint needs to consider, and the workflows between these viewpoints have been identified, a higher view diagram - Structuring Viewpoints in the System - helps show the whole workflow from all viewpoints and interactions. The Viewpoint Template and Viewpoint Service Template help organize the specific requirements. They
aid in identifying the rationale, specification, and nonfunctional requirements. This is a
great tool for making the requirements gathering process more systematic and uniform
between analysts.

Requirements in Agile Software Development
The software industry today is moving away from traditional methods such as the
Waterfall model and the Spiral model in order to be more flexible with the ever changing
requirements during the Software Development Lifecycle Process. More companies are
leaning toward “Agile Software Development” which tries to be more customer-oriented
and approachable. The Agile Manifesto states valuing “individuals and interaction over
processes and tools, working software over comprehensive documentation, customer
collaboration over contract negotiation, and responding to changes over following a plan
[9].”

According to Requirements Engineering in Agile Software Development [9], “the main
difference between traditional and agile development is not whether to do RE but when to
do it.” In traditional methodologies, requirements gathering is the very first step and stops
completely once entering the design phase. In contrast, agile methodology welcomes
changes to requirements at any point in the lifecycle, even quite late in the process.

The main goal of RE process is creating a system requirements document for knowledge
sharing, while Agile Development (AD) methods focus on face-to-face communication
between customers and agile teams to reach a similar goal [9]. There are several
techniques to requirements engineering in different agile methodologies. Some of these
include:

- Agile which aims to keep the number of models and documentation low and uses
  brainstorming and a requirements elicitation technique
- Feature-Driven Development which is a five-step process focused on building and
design phases
- Dynamic Systems Development Method focused on gathering the base
  requirements during the early feasibility study and business study phases and all
  other requirements during the development process
- Extreme Programming which elicits requirements as customer-written user stories
- Scrum which uses a Product Backlog system to track and push requirements
  forward by constantly refining and updating it during meetings and Sprint cycles

The bottom line of being Agile means being able to “Deliver quickly. Change quickly.
Change often” and thus, the agile RE activities are the Feasibility Study, Elicitation,
Analysis, Documentation, Validation, and Management [9]. The Feasibility Study
outlines the description of the system and its usability in the organization to allow for
achievability and viability assessment in order to evaluate whether continuing the project
is worthwhile. During Elicitation, “agile teams work with stakeholders to find out about
the application domain, the services that the system should provide, the system’s
operational constraints, and the required performance of the system (non-functional
requirements).” In Requirements Analysis, the gathered requirements are assessed for
completion and clarity, and the resolution for the lack thereof. Requirements
Documentation, though minimal in agile methodology, is still important as an immediate
communication tool for stakeholders and for future reference. Requirements Validation, checks to make sure that the requirements actually define the customer-requested system. And finally, the most important area of Requirements Engineering for this paper, Requirements Management, which entails understanding and controlling changes to system requirements by storing and prioritizing requirements, tracking development progress, and providing a level of requirements traceability [9].

**Application to Case Study**

As mentioned in the previous section, requirements management is one of the areas evaluated for the quality of a program’s management. Program managers can regard requirements as the contract between the developer and the customer on a program, and manage the customer’s expectations by managing the requirements [7]. Requirements management focuses on managing the process of extracting, developing, defining, and refining the requirements of a software program [7]. Taking into consideration the proposed company for which a case study would be conducted, it holds that as the program is already built with the most important requirements already executed, the only time requirements need to be extracted is when a customer requests a specific customization that the current default software does not already entail. As described in the section *Software Customization*, a software support member is responsible for understanding the business need of the customization and documenting the requirements accordingly, followed by receiving a quote from a developer as to how many programmer-hours are required to complete the project. This means that it is expected that the manager’s role entails developing, defining, and refining these requirements, or overseeing that these measures are taken if they are already done so by the software support team. The following are additional points stated by [7] which qualify as requirements management best practices and serve as the basis for the survey developed to assess the quality of management in the proposed company:

- Quality requirements management ensures that iterative and unanticipated changes are maintained throughout the project lifecycle.
- Quality requirements management must establish procedures and structure to ensure that requirements specifications are complete, consistent, readable, unambiguous, traceable to their origin, and do not arbitrarily contain design stipulation.
- Quality requirements management addresses the requirement attributes, including the following: managing customer benefit, the requirements author and/or responsible parties, the corresponding effort, the development priority, rationale, and relationships to other requirements. Quality requirements management is determined by the effort in tracking status, dates, and versions.
- Quality requirements management will facilitate the user/customer needs into requirements that can be implemented.

The following is thus, a proposed survey which would be distributed to the software support team and the project manager in order to be able to gather data to evaluate how the requirements engineering management can be improved. In the survey questions the responses to the "thought experiment" are shown in red.
**Requirements Engineering Assessment Survey**

Answer the following questions to the best of your ability and knowledge regarding the current Requirements Engineering (RE) process at your company. The questions are organized into sections based on the different stages of the RE process. Answer questions on a scale of 1 to 10, let 1 = Not At All, 5 = Neutral, 10 = Strongly Agree

**Feasibility Study**

When discussing the requested changes or additions to the system, you attempt to understand the business needs for the requested changes.

1 2 3 4 5 6 7 **8** 9 10

When discussing the requested changes or additions to the system, you help the customer prioritize their needs in order to help them stay within their budget and at the least, accomplish their top priorities.

1 2 3 4 5 6 7 8 9 **10**

**Requirements Elicitation**

Your approach to having an honest and helpful channel of communication with the customer is to hold an open interview, and allow brainstorming to take place if the customer needs help finding a solution to a business need.

1 2 3 4 5 6 7 **8** 9 10

When discussing the requested changes or additions to the system, you create a UML use case analysis together with the customer to identify the actors involved in a certain event.

1 2 3 4 5 6 7 8 9 10

When discussing the requested changes or additions to the system, you create a UML sequence diagram with the customer to help describe the steps of the event itself.

1 2 3 4 5 6 7 8 9 10
**Requirements Analysis**

When prepping for requested changes or additions to the system, you attempt to have all the actors affected by the changes participate in a Joint Application Development workshop to promote communication amongst each other to prevent conflicts as a result of the changes.

Before approaching a developer with the requested changes or additions to the system, you try to make sure that the requirements are clear, complete, unambiguous and don’t contradict.

**Requirements Documentation**

You prepare documentation to transcribe the requirements for the requested changes or additions to the system.

When (if) documenting the requirements, you use peer interviews to ensure the accuracy and quality of the documentation.

**Requirements Validation**

After collecting all the requirements, you participate in requirements validation by conducting requirements reviews to walk through each requirement checking for conflicts, errors, extras and omissions with the customers.

After the developer has programmed the customizations, you conduct acceptance testing by having the customer test the system and check that it satisfies the contractual acceptance criteria.

**Requirements Management**

You have a tool or process for requirements’ storage, prioritization, change and development tracking, and traceability.
Evaluation and Proposal of a Requirements Management Program

Based on the responses from the Requirements Engineering Assessment Survey, this section is a suggested proposal of a Requirements Management Program which could be implemented to improve the current Requirements Engineering process.

The Feasibility Study, Requirements Elicitation, and Requirements Analysis questions assess whether the software trainers are properly devoting time and effort to understanding the customer’s position and why they have a need which is not met with a solution provided by the software as it currently exists. Feedback from the Feasibility Study suggests that sufficient effort is made to understand the requested changes and help the customer prioritize their needs. The Requirements Elicitation suggests that the current process does not use in depth techniques like UML use cases and sequence diagrams for requirements elicitation. Management needs to implement a program which trains support team members on how to open up a channel of communication between the customers and other support members to promote discussion of customization feasibility, requirements elicitation, and requirements analysis. This line of communication is important for a number of reasons. First, it is important for customers to feel that their needs have solutions. The whole point of purchasing the system is to minimize their manual efforts and automate their workflow as much as possible. Second, it is important that more than just one team member be involved in the discussion to allow different experiences and approaches to reach optimal possible resolutions. It may be the case that the current software already handles the customer’s request, but because of inexperience or other reasons, the team member is unable to offer a solution and thus pushes the customization forward which requires developer hours, testing time, future maintenance and compatibility issues, etc.

It is also important for the support team member to be able to facilitate a Joint Application Development workshop for the customer so they have clarity amongst
themselves. It is important that one department on the customer’s end is not making changes to the program that may affect the workflow of other departments, especially if these effects conflict with their own processes. The training program also includes educating team members on how to incorporate use cases to help guide customers to describe their step-by-step workflow. This helps define requirements for the customized functions in the system. It is always better to find a solution within the existing version than constantly add changes specific to customers which need to be kept track of and documented to control future complications and version conflicts. By having management implement a training program and step-by-step process, assessing customer customization feasibility and requirements elicitation become smoother practices.

The Requirements Validation questions assess whether software support team members know how to conduct proper reviews before and after the customizations have been developed to make sure that the requirements are creating the right functions needed by the customer and that after development, the customizations are reviewed by the customer for acceptance satisfaction. The responses to this section indicate that proper requirement validation steps are not taken prior to implementation. Acceptance testing is conducted after the customization is created and implemented but the cost of finding mistakes during acceptance testing is much higher than finding them during requirement reviews prior to any programming changes.

The Requirements Management questions assess whether there is a proper tool or process for documenting requirements. Marks are low for the entire section of Requirements Management. Management is encouraged to invest in project team sharing tools that allow the requirements to be documented throughout their lifecycle. These documentation notes are publicly available to team members, allowing members to pick up where someone left off, use them for peer review, or allow their use by management. Proper documentation is also critical for allowing team members to be able to have a log to reference in the future. It is highly encouraged that documentation has a standard procedure so all requirements are documented in a unified format to homogenize paperwork making present and future work more seamless and transparent. It is also important that the team has a manager who has the ability to identify team members’ strengths, and oversee that work is distributed respectively ensuring that these strengths are properly utilized. No one member of the team should have to take the burden of the entire process by him or herself. By implementing these recommended tools and processes, management should be able to use the available data to oversee the process of each customization project and intervene where they see road bumps.

**Measure of Success**

To reiterate what was previously mentioned, getting requirements right might be the most important and difficult part of the software project, and deficient requirements are the single biggest cause of software project failure [10]. This section was devoted to summarizing the RE process of a software development lifecycle and coming up with a survey to assess the RE process of the proposed company. A plan of action was also proposed, which may be adopted by project managers based on the results of the assessment survey. The final step of this section is to propose a measure of success. If the proposed company were to implement the suggested requirements management program...
based on the proposed assessment survey the following describes how to measure success.

Several approaches can be combined to measure success. One approach can be to collect quantitative data before, during, and after the implementation of the new requirements management program. The data should include the following:

- The planned and actual duration of customization projects from the beginning feasibility study to the end after implementation and acceptance testing
- The number of planned and actual person-hours spent on an RE cycle, defined as a set of activities that contain at least one each of elicitation, modeling, validation, and verification activities [10]
- Planned and actual costs
- The number of team members involved
- The number of iterations the stakeholders had to go through to reach a successful implementation of the customization which solved their initial business need
- The number of defects produced by the customization

By analyzing this quantitative data, it can easily be seen if there are RE process improvements after implementing the suggested requirements management program. Improvements are indicated by seeing a smaller gap between planned and actual duration of customization projects, a smaller gap between the number of planned and actual person-hours spent on an RE cycle, a smaller gap between planned and actual costs, a greater number of team members involved, a smaller number of RE cycle iterations, and a smaller number of reported defects.

The second approach is to collect qualitative data. Stakeholders can rate the quality of RE products using these quality attributes: correct, unambiguous, complete, consistent, prioritized, verifiable, modifiable, and traceable [10]. Qualitative data analyzing performance is also an important factor, evaluated by measuring quality of RE service, quality of RE products, and process control. Measure of success using these factors as metrics should indicate reported improvements in each of the quality attributes and performance quality attributes.

The suggested RE management improvement program is based on best practices exhibited by the most successful RE teams [10]:

- Involve customers and users throughout RE to better understand “real needs”
- Identify and consult all likely sources of requirements to improve requirements coverage
- Assign skilled project managers and team members to RE activities for more predictable performance
- Allocate 15 to 30 percent of total project effort to RE activities to maintain high-quality specification throughout the project
- Provide specification templates and examples to improve quality of specification
- Maintain good relationships among stakeholders to better satisfy customer needs
- Prioritize requirements to focus attention on the most important customer needs
• Develop complementary models together with prototypes to eliminate specification ambiguities and inconsistencies
• Maintain a traceability matrix to maintain an explicit link between requirements and work products
• Use peer reviews, scenarios, and walk-throughs to validate and verify requirements

Success is measured by the noticeable improvement in the proposed company by implementing the listed recommendations.
Section 5: Risk

Overview of Risk Management
Almost every literature related to software risk management begins with an introduction about how risk management is the most trivialized or avoided area of software development. Kwak and Ibbs identified risk management as the least practiced discipline among different project management knowledge areas [11]. Different reasons are provided as to why most software organizations shy away from it. Some examples are that most software developers and project managers perceive risk management processes and activities as extra work and expense [11]. Another states that knowledge about possible risk management methods and tools has not reached most practitioners [12]. Many existing risk management approaches have both practical and underlying, theoretical limitations that hinder the usability of these methods [12]. And finally, that while there are several anecdotal descriptions of managing risks in practice, there are few reports on systematic and scientifically sound evaluations to provide empirical feedback on their feasibility and benefits [12]. But the fact still remains that although it might be hard to manage, risk analysis is a critical part of the software development process which needs to be taken into consideration when taking on projects in order to be able to manage their costs, schedules, and requirements. Without taking risks into consideration, managers are estimating budgets, merely guessing timelines, and taking on project scopes that might very well not be feasible with the present constraints. Effective risk management is the most important management tool a project manager can employ to increase the likelihood of project success [11].

To start with, the following are risk management definitions provided by [12]:

- Risk is a relative concept and it is always dependent on the goals, expectations and constraints involved
- Goal is a general statement of purpose, direction or objective
- Risk factor is a characteristic that effects the probability of a negative event occurring
- Risk event represents the occurrence of a negative outcome
- Risk reaction describes a possible action that can be taken as a response to a risk event
- Risk effect represents the impact of risk event-reaction combination to project goals
- Utility loss captures how severe the overall impact of effects has been to different stakeholders

For software managers seeking decision-support for quantitative risk management, tools based on industry-wide empirical studies and benchmarking data can certainly provide a vast improvement over using gut instinct and heuristics [13]. One example of such a tool is Bayesian belief nets (BBNs) model, graphical networks together with associated sets of probability tables, which can be executed using recent algorithms and software tools like Hugin [13]. BBN models can be thought of as risk management decision-support tools that build on the relatively simple metrics that are already being collected. Per the
literature in *Software Metrics: Roadmap* [13], it is by far the best solution for a model which can handle the different risk factors as well as consider uncertainties. By combining different types of information and with the ability to forecast despite missing data, such a model can predict general software quality attributes like defect-density and cost among a long list of other information it can provide.

It is important that the project manager alone is not the sole contributor to risk assessment. Team members should be educated on how to use risk management tools and include risk management as part of their everyday process. It is not enough, or even recommended to have a separate risk management team. The goal is to try to have project team members incorporate risk management practices to their routine project activities.

Although risk management is a harder undertaking than most other areas of the software development cycle, the benefits from molding a company culture to accept it, can make the difference between success and failure. A company which embraces risk management in their daily activities, takes the necessary precautionary measures to eliminate the highest risk factors, which usually stem from human nature itself. Finally, effective risk management evolves with the project and its continuous changes and takes into account the long term risks over immediate objectives.

**Application to Case Study**
Looking back at the proposed metrics for assessing quality of software management for the proposed company, risk analysis is one of the four categories being evaluated alongside requirements engineering, people management, and process management.

Following the GQM paradigm, the assessment survey was created to meet a certain software management goal: to implement an effective risk management process. The purpose is to evaluate the Software development process in order to improve it with respect to the cost assessments from the point of view (perspective) of the manager, assuming the environment is a failing business model encompassing over-budget projects with insufficient risk assessment capabilities. The questions in the assessment survey were based on the causal models supported by [13], and the principles of software risk factors presented by [11]. In the survey questions the responses to the "thought experiment" are shown in red.
Risk Analysis Assessment Survey

Answer the following questions to the best of your ability and knowledge regarding the current Risk Management (RM) process at your company. The questions are organized into sections based on the different stages of the RM process. Answer questions on a scale of 1 to 10, let 1 = Not At All, 5 = Neutral, 10 = Strongly Agree

Data Tracking

There is available data from different testing phases, such as information about defects.

1 2 3 4 5 6 7 8 9 10

There is available empirical data from previous process and resources, which may be used as reference for similar projects.

1 2 3 4 5 6 7 8 9 10

There is subjective data referencing process and resources i.e. the quality and experience of staff.

1 2 3 4 5 6 7 8 9 10

Models

When assessing risk, management implements a causal model. For instance the thought process is “Y is caused by X” versus “Y is affected by X.”

1 2 3 4 5 6 7 8 9 10

The model used to assess risk can handle diverse processes and product variables.

1 2 3 4 5 6 7 8 9 10

The model used to assess risk can handle empirical evidence and expert judgment.

1 2 3 4 5 6 7 8 9 10

The model used to assess risk can handle uncertainty.

1 2 3 4 5 6 7 8 9 10

The model used to assess risk is effective even without complete information.

1 2 3 4 5 6 7 8 9 10
## Tools

Communication tools are used to explicitly define different aspects of risk.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>

Software tools are utilized to implement the model in use. For example, the tool can generate probability tables to relieve the manual labor of working with complex risk analysis algorithms.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>

## Risk Analysis

Risk has an operational definition on a detail level during risk analysis.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>

Risks associated with inaccurate estimating and schedule planning are considered.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>

Risks associated with incorrect and optimistic status reporting are considered.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>

Risks associated with external pressures, which damage software projects are considered.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>

Personnel shortfalls are taken into consideration during risk analysis.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>

Development of wrong functions and properties are taken into consideration during risk analysis.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>

Developing the wrong user interface is taken into consideration during risk analysis.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>

Gold plating (adding more functionality/features than is necessary) is taken into consideration during risk analysis.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
</table>
Risk of requirements changing is taken into consideration during analysis.

```
1 2 3 4 5 6 7 8 9 10
```

Real-time performance shortfalls are taken into consideration during risk analysis.

```
1 2 3 4 5 6 7 8 9 10
```

The probability of risk effects and utility losses are estimated for all risk scenarios.

```
1 2 3 4 5 6 7 8 9 10
```

**Risk Control Planning**

Risks are ranked based on their probability and utility loss for each stakeholder.

```
1 2 3 4 5 6 7 8 9 10
```

The highest ranking risks are used for control planning and control actions.

```
1 2 3 4 5 6 7 8 9 10
```

**Risk Prevention**

After risks have been identified and quantified, measures are taken to reduce the probability of high risk elements by taking proactive measures to reduce the risk exposure.

```
1 2 3 4 5 6 7 8 9 10
```

**Evaluation and Proposal of a Risk Management Program**

Based on the responses from the Risk Analysis Assessment Survey, this section is a suggested proposal of a Risk Management Program which could be implemented to improve the current program, if any. When administering and evaluating the survey, it is important to recognize the needs for risk management by the proposed company. Much like in the Requirements Engineering section, this section also targets the risks involved with customization projects requested by customers. Risk analysis could show if taking on a certain customization project is going to be profitable for the company based on the existing constraints. Along with assessing risks involved with the new system features or functionalities programmed for customizations, risk also plays a role in the company’s decisions to make any general, default software changes to keep up with changing industry requirements. These changes could be anything from new technology introductions into the business flow or changes in the laws or standards of the target industry which the software is geared toward. Risk assessment in this area could define differences between competitors and attract new customers toward one software solution over another based on these updates in industry and software technology.
The assessment survey is divided into the following sections: Data Tracking, Models, Tools, Risk Analysis. The statements in “Data Tracking” first assesses that there is no data being tracked in various aspects of development and resources such as testing and quality of staff. It is important to take this type of data into consideration when doing risk analysis, as all these factors affect the probabilities of outcomes, whether they be negative or positive. Data is therefore the fundamental element for risk assessment, without which there is really no starting point. Related to this idea is the section on the assessment survey regarding Tools. Although the above mentioned data can be collected manually, a smaller company as the one proposed will probably not want to waste human resources on tracking information like defect rates and quality of staff. Instead, to make risk management easier and more affordable, it would be encouraged to invest in automatic data tracking tools.

The second section checks for the implementation of a causal model. This implementation choice is based on the arguments made from literature review of Software metrics: roadmap [13]. According to their research, the hope is that the risk model implemented (if any), is not “the more common regression model.” The regression model works by having the input be a function of the output instead of vice versa. If a causal model was being used, then the last four statements of the “Model” section in the survey would assess what variables the model can handle. Since no model is used in the Proposed Company, a risk management program needs to be implemented from the ground up starting with a causal model. This is because the causal model can handle diverse process and product variables, empirical evidence and expert judgment, genuine cause and effect relationships, uncertainty, and incomplete information [13].

The third section of the assessment survey gathers information about the usage of any tools that would assist in risk management. The first statement in this section is to evaluate how definitions of risk analysis are communicated. The low marks in this section means tools like the Riskit Analysis Graph [14] and software tool for risk analysis may be introduced in the improvement program. Riskit is a comprehensive risk management method that is based on sound theoretical principles. It has been designed to have sufficiently low overhead and complexity so that it can be used in real, time-constrained projects [14]. The Riskit Analysis Graph is used to define the different aspects of risk explicitly and formally by decomposing risks into clearly defined components, or risk elements [12]. The graph starts with risk factors which may affect one or more risk events. Risk events usually fire risk reactions. Goal effects represent the impact the risk reactions had on each goal, and the utility loss simultaneously considers multiple criteria capturing how severe the overall impact of effects has been to different stakeholders [12]. Software tools are important since the algorithms involved can only be efficiently implemented with the use of specific risk management software.

The fourth section of the assessment survey is Risk Analysis. The first question reinforces the concept first assessed in the Tools section: whether risk is clearly defined, since the most important foundation for risk analysis is to have clearly stated definitions for risk. The following three statements are based on key software risk factors and concerns presented by Jones for executives and software managers [11]. The next nine statements are selected software risk items presented by Boehm [11], which should be considered during risk assessments. The majority of these statements scored low on the
survey and need to be considered in the risk management improvement program. The last two statements of the section are based on steps identified in the Riskit method [12] for risk analysis. Per the Riskit method, risk analysis should categorize risks, complete risk scenarios for each risk event, estimate risk effects, and estimate probabilities and utility losses of risk scenarios. This may be done by using a Riskit Pareto ranking technique, which is a two-dimensional space to position risk scenarios by their relative probability and utility loss [14].

The Risk Control Planning section is also created based on the Riskit “risk control planning” step, which is used to design selected risk controlling actions as the input to management’s efforts to use the risk data to implement programs or company policies to decrease high risk exposures. For example, Boehm and DeMarco showed that the probability component of risk exposure for employee turnover (one of the highest risk elements of most software projects) can be reduced by: empowering performers, teambuilding, establishing significant incentive bonuses for successful project completion, etc [11]. This section’s score was mediocre. The improvement program would include a section geared towards utilizing the collected data from risk analysis processes to generate management decisions for lowering risk exposures, which also happens to be the last step in the Riskit method for “controlling of risks” with the intent of reducing risks.

**Measure of Success**

Although *Empirical Evaluation of a Risk Management Method* [12] explains why evaluating risk management methods are problematic, some intuitive measures of success are presented here. When assessing the risk management for the proposed software company, it is important that the company has ways not only to collect and manage data throughout the software development cycle, but ways of collecting the right type of data which will be of value when it comes time to assessing and reducing risks. The most significant benefit of software metrics is that they are supposed to provide information to support quantitative managerial decision-making during the software lifecycle – good support for decision-making implies support for risk assessment and reduction [13].

To measure success after implementing the suggested Risk management improvement program, the company can be evaluated against the following baseline for risk management best practices provided by *Project Risk Management: Lessons Learned from Software Development Environment* [11] to see how closely management has come to carrying out these principles:

- There is a shared vision for success based upon commonality of purpose, shared ownership, and collective commitment
- There is effort put into thinking toward tomorrow, anticipating potential outcomes, identifying uncertainties, and managing program resources and activities while recognizing these uncertainties
- Open communication allows for free flow of information at and between all program levels through formal, informal, and impromptu communication and consensus-based processes
- Individual perception is valued, which can bring unique knowledge and insight to the identification and management of risk
• Risk management is integrated into management as a vital part of program management
• Proactive strategies are applied that involve planning and executing program activities based on anticipating future events
• A systematic approach that is adaptable to the program’s infrastructure and culture is instated
• A continuous process characterized by routine risk identification and management activities throughout all phases of the life cycle of the program is in place

If the proposed company demonstrates most or all of the above mentioned principles then it can be stated with confidence that the risk management improvement program was a success.
Section 6: Planning/Estimation

Overview of Time Management
Time plays a crucial role in any type of business and project-oriented environment. It is particularly delicate in software engineering, as the software development lifecycle does not progress on a linear timetable. Trying to resolve scheduling delays is complicated as well. As Brooks writes in *The Mythical Man-Month* [15], “when schedule slippage is recognized, the natural (and traditional) response is to add manpower. Like dousing a fire with gasoline, this makes matter worse, much worse.” Trying to pick up where someone left off has a very steep learning curve in software projects so replacing human resources or trying to add help to an already behind-schedule project only makes matters worst. As *The Software Engineering Timeline: A Time Management Perspective* [16] observes, many of the problems associated with software projects, such as planning, time pressure, and late deliveries, are time related. For all these reasons and more, time management, planning, and estimation is a large part of management’s responsibility.

Time management was not an integral part of the early software engineering industry around 1970. From the literature review done by [16] “time” was almost never mentioned in traditional software engineering books and “schedule” was used only sparsely. In more recent literature covering organizational processes which use the traditional waterfall method at the turn of the millennium, time was referenced in the context of “time to market” but again, not in relation to planning, scheduling, and meeting deadlines. Per the investigations and data collected from surveys at different software companies which use the traditional waterfall method, time management came up regularly as an issue and obstacle for timely and realistic project deliveries. Example suggestions for improving software quality were, “allocation of time to learn the things [to be implemented] before we rush to the next coding; allocation of enough time for review and debugging,” “allocation of time for design and education in design topics,” and “add time to the development and testing [stages] [16].” The review done covering software processes in the 1990's shows more attention being put into time management. Such processes include Team Software Process (TSP) and Rapid Application Development (RAD), both of which took managing time into account, with RAD even advocating meeting the agreed deadline at the expense of functionality. More recent processes reflect the fact that time is becoming more of a critical factor in recent years. Such processes include, for example, agile development and extreme programming which place importance on project planning and time estimating. Agile assumes time, resources, and quality as fixed variables with scope being the only variable allowed to change. It introduces time boxing which is a tool used to force customers to make decisions for the short-term direction of the project based on near-term goals and to force the team to deliver useful outputs in short, defined periods.

In general time management is useful to keep teams on track and deliver products on-time. It also helps avoid these common time-related issues [16] lists. Bottlenecking occurs when a team member(s) has no work to do at a given time in the development process because their work depends on the output of a previous team member(s) who has fallen behind schedule. Feasible project planning and scheduling are common obstacles
when enough time is not allocated to certain parts of a project causing difficulties meeting scheduled deadlines. Time pressure ensues because of the aggregate of the previous issues as deadlines approach and faulty time estimations and bottlenecking puts pressure on team members to reach scheduled goals. And finally as a result of any of the time management issues already mentioned late delivery results.

Implementation of Time Management

To implement time management, a scheduling tool needs to be considered. A well-known and widely used technique in management since the 1950s is the Critical Path Method (CPM). It was originally formulated to help regulate the amount of time each task in a maintenance and construction project took and try to minimize the length for routine and repetitive tasks [17]. The CPM breaks up the project into individual tasks and analyzes the dependencies between these activities. There are some tasks that require the output of a previous task and thus cannot begin without the completion of that task or collection of tasks, thus creating a “path” representing the progression of the project as a train of dependencies. The formal definition of Critical Path as provided in Business Management and Economics Sections [17] is “the sequence of project network terminal elements with the longest overall duration, determining the shortest time to complete the project.” The critical path is realized when critical tasks are dependent on each other due to technological constraints. A critical sequence is realized when critical tasks are dependent on each other due to resource constraints. By having this visual tool, it becomes clear how long each task will take, and in turn, how long the project as a whole will take. The following is a summary of CPM guidelines:

Several underlying assumptions are made in the CPM approach. The major ones are that a project can be broken down into a series of identifiable tasks, each of which may also be further broken down into subtasks. Once this breakdown has been accomplished, the tasks are then placed in order against a timeline. Each task is assigned a start date, duration, and end date, and may also have various resources attached to it. These resources can include specific personnel, a budget, equipment, facilities, support services, and anything else that’s appropriate. The common way to perform the task is to draw the tasks as horizontal boxes against a vertical time scale. The resulting chart is called a Gantt chart [17].

There are two ways that the Critical Path can be identified: one is referred to as the forward pass, and the other the backward pass. The forward pass attempts to evaluate the earliest completion date for a project by lining up tasks according to the earliest starting date of the project. The backward pass starts from the date the project needs to be complete and works backwards to establish the late start dates all the way to the initial task.

To analyze the critical path the project undergoes the phases for Planning, Analysis, and Scheduling and Controlling. By implementing these phases most aspects of the project get considered, including time, cost, and resources. By knowing when certain tasks are coming up, management can make sure that resources are available as needed and if they are not available, having the budget to acquire them in time. It is recommended that an organization uses this technique to implement standard operating procedures (SOP) for tasks that are similar in nature and executed more frequently. This way a baseline is
created for which to measure progress against. It also creates a wealth of information to reference back to during future operations thus helping upcoming project timelines and estimations. The more projects are documented, the clearer it becomes how much time and effort, and in turn cost, each task consumes. Of course, just having this information and creating a project timeline does not guarantee success. Projects undergo unforeseen changes all the time including requirements changes, budget cuts, and people coming and going, all considerations management needs to be aware of to do its job efficiently.

**Application to Case Study**
The following section takes a look at implementing time management using the Agile software development methodology.

*Designing an Agile Development Environment for Time Management*
Previously mentioned in the section for “Role of Time Estimation, Planning and Scheduling,” literature review done on organizations using the traditional waterfall methodology especially in the early days of software engineering, did not consider time to be of critical concern. As the industry evolved and time factored into the development lifecycle, new methodologies took shape. In the present day industry, Agile development seems to be the methodology being practiced by those companies which take process compliance seriously and are actually pressured by time constraints. “Since it is difficult to track software time dimension, it must be developed through a tight process. Agile software development is an example of such a process that systematically supports time tracking of software projects [16].” For this reason, the stance this study takes in the area of time management is based on the Agile development methodology.

The reason Agile development is so closely related to time management is because in design, it was developed to keep a fast paced and forward-moving environment. It focuses on rapid turnaround, incremental planning, customer involvement, and continuous integration all of which mitigate risk and procure usable software in short time increments [18]. The process works by attempting to yield a manageable set of functions in cycles which usually last between one to four weeks. Daily and weekly meetings keep all team members up to speed on the status of each function and customer involvement ensures validity. Agile methodology often incorporates pair programming which has two developers working on pieces of code together on the same computer for reinforcement of code logic and structure. The survey developed for the proposed company is not geared toward pair programming though since the size and structure of the proposed company would not be able to sustain this technique. Otherwise, the survey developed is consistent with the Agile choice of management style and the use of the Critical Path Methodology for time estimation and task planning. In the survey questions the responses to the “thought experiment” are shown in red.
Scheduling/Planning Assessment Survey

Answer the following questions to the best of your ability and knowledge regarding the current Time Management (TM) process at your company. The questions are organized into sections based on the different stages of the TM process. Answer questions on a scale of 1 to 10, let 1 = Not At All, 5 = Neutral, 10 = Strongly Agree

Time Management

There is importance given to time estimation.

1 2 3 4 5 6 7 8 9 10

There is a disciplined and logical approach to planning, scheduling, and management of projects.

1 2 3 4 5 6 7 8 9 10

Time-cost relationships are considered when evaluating the implementation of a new project.

1 2 3 4 5 6 7 8 9 10

Critical Path Method

* The Critical Path Method is defined as the sequence of project network terminal elements with the longest overall duration, determining the shortest time to complete the project.

Graphical drawings are used to breakdown tasks by identifying various activities that need to be accomplished for project completion.

1 2 3 4 5 6 7 8 9 10

These drawings or task breakdowns are used to find the project’s Critical Path* (or similar schedule of tasks) which helps identify risk factors and properly allocate resources.

1 2 3 4 5 6 7 8 9 10

These drawings or task breakdowns allow estimations of project completion times.

1 2 3 4 5 6 7 8 9 10

Slack or float times are identified to manage the reallocations of resources.

1 2 3 4 5 6 7 8 9 10
The Critical Path* or similar time management tool is updated regularly to reflect the various stages of the project completion.

Project schedules and timelines are documented for future reference to help identify time estimates in future similar projects.

**Agile Process**

Tasks are prioritized, defined, and assigned.

The task breakdowns are organized into cycles with manageable deadlines.

There is practice of holding regularly scheduled meetings to review the status of tasks and assign a new set of tasks for the upcoming week.

The development cycle spans the defined time frame. For example, at the beginning of the project, the length of the development cycle is set at four weeks.

**Tools**

There is a task tracking system, to keep the task statuses up-to-date and public.

There is a build management tool to run tests on new functions and notify developers or responsible parties for test failures.
Evaluation and Proposal of a Time Management Program

The assessment survey for time management has been divided into four sections: Time Management, Critical Path Method, Agile Process, and Tools. The questions in the first section are to assess if time and scheduling are considered at all in the proposed company. From the description of the company provided in the beginning of the paper, the proposed company does not include time management into any development process considerations. For this reason, the time management program proposed, includes the statements in the Critical Path Method section of the assessment survey. It is suggested that management uses visual tools like drawings to breakdown activities into tasks to estimate completion times, identify slack times, and update project progress.

Since Agile supports incremental planning with manageable tasks, the following section can use the tasks identified from the Critical Path method to create an Agile development environment. When applying this idea onto the proposed software company, the functions being created are mostly the customization projects in addition to any new functionalities seen as important for the general default release of the product. These features could include “nice to have” features deemed useful for all clients using the product, or changes being made due to industry standard and protocol changes or in keeping up with the advances in technology.

Agile is also big on involving customers in the development process, which highly applies to the proposed company, since the software is geared toward meeting the business needs of a niche industry, no one knows the requirements better than the customer. Management needs to make sure their involvement is strongly emphasized and encouraged. Unlike the waterfall method which “falls into the predictive category because the entire development activity including the milestones is determined at the beginning of the software project [18],” the Agile method is better suited for the proposed software company because the software in question is not being created from the ground up. There are no significant milestones. The software requirements are smaller projects which modify or add to the existing product, making the nature of Agile compatible with this reality.

What is perhaps the most important contribution of Agile for the proposed company, is the practice of holding meetings. Traditionally in Agile, a cycle commences or concludes with a weekly planning meeting. JPL conducts their meetings as described in A Roadmap for Using Agile Development in a Traditional Environment [18], “The meeting begins by reviewing the status of each of the detailed tasks that had been planned for the previous week. The tasks are displayed on a screen using the issue tracking system (JIRA). Each developer has recorded the progress of the previous week. It is common and expected that not all tasks planned are completed; a few tasks than can actually be completed are included each week in case other assigned tasks are completed earlier than predicted or must be postponed because of some unexpected barrier.” This characteristic of Agile to have such weekly face-to-face meetings is a missing step in the proposed company’s development process. Without such meetings, team members are completely oblivious to each other’s projects. If someone were to have to take over another person’s project, they would have to start the development cycle from the beginning, starting with going over the requirements with the customer to even know what the business need is. As described, the proposed company has no system for tracking project progress or
consulting with each other through meetings. This causes disconnect between members and results in less than optimal software solutions. If customer customization projects were discussed openly amongst members, brainstorming and other activities could help come up with better solutions than if only one person was responsible for its implementation from start to finish. Experience can make a big difference when it comes to problem solving.

The last section of the assessment survey addresses development process tools used in the Agile environment. An issue tracking system like JIRA (https://www.atlassian.com/software/jira), is an example of this type of tool to consider. This makes all issues public and open for discussion. It could be the case that two different customer cases have the same unmet business need that requires a functionality solution. If two different team members are dealing with the cases and neither is aware of the other’s problem, this could result in double the development work if different programmers were involved or the problem was proposed to a developer in different ways, creating two different functionalities where one solution would have sufficed. This type of cluster makes tracking hard and increases the amount of knowledge software trainers need to have. Another tool as suggested by [18], is Confluence (https://www.atlassian.com/software/confluence), a server software used to create and modify web page content, which would be used to document meeting notes, design decisions, diagrams for user requirements, and other such “documentation.” And finally, incorporating a build management tool like Cruise Control (http://cruisecontrol.sourceforge.net/), to make update releases more seamless and tested. As it stands, the proposed software company does not take advantage of any of the suggested tools, which makes all program changes dependent on the manual labor of specific team members instead of dependent on tools and processes. This creates slack time for individuals waiting on each other’s outputs, delaying product deliveries, and lowering customer satisfaction.

Measure of Success
The goal of time management implementation would be to have a more efficient and productive development cycle. With the given suggestions based on the assessment survey, the proposed company’s way of planning should evolve to a more systematic and predictable process. Measures of success would include measurable timetables of tasks through such methodologies as the Critical Path method, individuals’ awareness of each other’s projects due to daily or weekly team meetings, documentation through the use of task tracking software, more integrated executable update capacities through the use of build management tools. The ultimate goal is to try to reduce risks by creating processes which are self-sufficient without being dependent on the skills or outputs of specific team members, “Specifically, agile software development renders the development processes more transparent and thus supports the development process tracking [16].”
Section 7: People

Overview of People Management

PMBOK [2] – *A Guide to the Project Management Body of Knowledge* – refers to the people management section as Human Resource Management, “Project Human Resource Management includes the processes that organize, manage, and lead the project team. The project team is comprised of the people with assigned roles and responsibilities for completing the project [2].” It breaks up the process into four well-defined and distinct categories: develop human resource plan, acquire project team, develop project team, and manage project team.

Developing the human resource plan requires identifying the organization’s activity requirements to determine the type of people needed for positions, and work environment factors such as culture, structure, and policies and being able to develop a plan using organizational tools such as charts and position descriptions. “The human resource plan, a part of the project management plan, provides guidance on how project human resources should be defined, staffed, managed, controlled, and eventually release [2].”

Acquiring a project team entails dealing with existing positions, negotiating arrangements to fill needed positions terms and agreements. Without the proper human resources, projects are subject to missing deadlines, going over budget, upsetting customers, and increasing risks. Developing the project team includes any factor that goes into enhancing project performance. These factors include team member’s skills, communication, and culture, “The basic work unit in innovative software organizations is the team rather than the individual. Such teams consist of `a small number of people with complementary skills who are committed to a common purpose, set of performance goals, and approach for which they hold themselves mutually accountable [19].’”

And finally, the goal of a well-managed project team is to optimize project performance. This entails performance-tracking, providing feedback, problem-resolution, and adaptability to change [2]. Keeping in line with the times, this paper turns toward the Agile development process as the optimal model for the Proposed company. As such, Agile’s stance on project team management states that “Agile companies practice leadership-collaboration rather than command-control management. They set goals and constraints, providing boundaries within which innovation can flourish. They are macromanagers rather than micromanagers. They understand that agility depends on trusting individuals to apply their competency in effective ways [20].”

Application to Case Study

The assessment survey for evaluating the quality of the Proposed company’s people management is developed based on the literature review of *Agile Software Development: The People Factor* [20] and *Impact of People Management Practices on Organizational Performance: Analysis of a Causal Model* [21]. As with the earlier section covering Requirements Engineering, this section is also based on the principals of the Agile Development Methodology. Therefore many of the statements included in the assessment survey are geared toward an agile environment. As with the previous sections, the survey
follows the GQM paradigm to meet the following software management goal: to implement an effective people management program. The purpose is to evaluate the proposed company’s human resource management model in order to improve it with respect to its effectiveness from the point of view of the manager, assuming the environment is not optimal when considering people factor. In the survey questions the responses to the "thought experiment" are shown in red.

**People Management Assessment Survey**

Answer the following questions to the best of your ability and knowledge regarding the current People Management (PM) process at your company. The questions are organized into sections based on the different stages of the PM process. Answer questions on a scale of 1 to 10, let 1 = Not At All, 5 = Neutral, 10 = Strongly Agree

**Competence**

The employment selection process is a rigorous process including written tests, programming tests (if relevant), and interviews.

1 2 3 4 5 6 7 8 9 10

The work environment is friendly and conducive to learning from others.

1 2 3 4 5 6 7 8 9 10

Performance appraisal reviews current level of knowledge and identifies gaps, upon which necessary steps are taken to enhance competence.

1 2 3 4 5 6 7 8 9 10

**Teamwork**

There is formal initiation or socialization upon induction into the company.

1 2 3 4 5 6 7 8 9 10

Projects are team-based.

1 2 3 4 5 6 7 8 9 10

Members of teams are placed in physically close proximity to one another.

1 2 3 4 5 6 7 8 9 10
Communication between team members is welcomed and encouraged to promote the exchange of valuable information.

1  2  3  4  5  6  7  8  9  10

**Organizational commitment**

You feel well-compensated for your work efforts.

1  2  3  4  5  6  7  8  9  10

You feel that you play a part in performance evaluations, and in fairness and justice of your organization’s policies.

1  2  3  4  5  6  7  8  9  10

You feel a sense of commitment to the organization based on growth opportunities.

1  2  3  4  5  6  7  8  9  10

**Customer orientation**

Performance appraisal by the customers is available.

1  2  3  4  5  6  7  8  9  10

---

**Evaluation and Proposal of a People Management Program**

Based on the survey, the weakest part of the overall Quality management, presented in this paper as the sum of Requirements management, Process management, Risk management, and People management, is the People Management of the Proposed Company. The assessment survey is divided into four sections: Competence, Teamwork, Organizational Commitment, and Customer Orientation. The proposed People management program includes all of the following recommendations based on responses to the statements of the assessment survey.

The section for “Competence” begins with hiring skilled human resources. The recruiting and hiring process is a critical factor for assessing “learn-ability, a candidate’s ability and willingness to learn.” Competent employees create quality software which shape a positive image of the organization establishing the groundwork for financial success [21]. More effort can be put into the hiring process which include tests to evaluate competence. A friendly work environment is also conducive to learning from peers, therefore the proposed company should consider promoting informal communication among workers to create bonds and free flow of information.

Teamwork begins with an “initiation or socialization” program. This foundation is missing in the Proposed Company, which is an important step to “sow the seed of teamwork and customer orientation that has to be matured in the continued interaction
with organizational members and events. A value-based induction process touches upon the organization’s mission, values, business details, customers, the expectations of the employees from the company and this in turn helps to build up an emotional bond with the company and members [21].” Team-based projects have many advantages. In team-based job design, the efficiency of the project teams allows employees to learn more from the team members than from formal training, significantly increasing the speed of delivery of service or product. Team-based job design enhances the teamwork of the members and facilitates more open discussion, encourages informal communication, and strong working relationships among employees. They develop a sense of group identity and willingness to help each other. The importance of this style is higher employee retention, product quality, and customer orientation [21]. The advantage of introducing an Agile environment is to “reduce the cost of moving information between people, and reducing the elapsed time between making a decision to seeing the consequences of that decision [20],” hence the improvement program suggests reforming the communication mode to invite more approachable, in-person and team-based exchange of information.

Assessing compensation satisfaction falls under “Organizational Commitment” since studies show using compensation to improve productivity of the organization, may not increase operating costs since compensation and rewards are major deciding factors for employee commitment [21]. Being able to feel like the employees have a say in the performance evaluation and justice system of the organization increases organizational commitment. The proposed company needs to change their company culture to have a more inclusive environment, where employees feel that their opinions are heard and make a difference, and where there is consistent feedback to their future growth opportunities and career development.

“Customer orientation” is also established by performance appraisal. This idea is especially particular to the Proposed Company because of the nature of its business model which includes a lot of customization projects. “In a customized software development organization, customer feedback on the product or service is part of the appraisal system. An appraisal system that incorporates the concerns of the customers becomes an effective tool to nurture a customer orientation [21]."

**Measure of Success**

Per the Proposed Company’s description, the current state of the company does not consider people management – or as referred to by PMBOK, human resource management – in too much depth. The measure of success in this section has a pretty low standard and any effort to improve in the above described criteria with positive results demonstrates successful improvement.

Progress based on the evaluation and proposal of a people management program should help develop the human resource plan, improve team arrangement and teamwork resulting in higher customer satisfaction, and strengthen organizational commitment as a result of improved team management.
Conclusion

The goal of this graduate project aimed to demonstrate clear understanding of the software development process and software management by creating a “thought experiment” which exercised this knowledge in a case study. Application of understanding is presented by the ability to describe a type of software company in detail, develop a new metric to measure management, apply this metric to the software company, and be able to propose improvement plans based on results of the metric.

Project management is an extensive field with many responsibilities to be considered. With the hypothetical software organization introduced in the beginning of this graduate project, four areas of focus were chosen for evaluation consideration. A metric was developed using the GQM approach to measure these areas of management.

For the proposed company, the areas of management chosen were requirements since the company frequently deals with customization projects; risk, since there are risk factors involved in taking on customization projects and dealing with customers so closely; time management, a critical factor in most software organizations; and people management based on the fact that the proposed software company is a small to mid-size company making human resource a valuable asset.

The benefits gained from going through this project include demonstrating the ability to apply knowledge of the software development process and software management, exercising development of tools for management evaluation, and showing analytical skills to develop courses of improvement.

The significance of this work is the presentation of a methodology for the evaluation of software practices which can be replicated, expanded, and implemented in the real world. The metric proposed in this graduate project was created specifically for the company described earlier in the paper. Future studies or bodies of work can use the surveys provided to evaluate similar companies as the one proposed here, or using the same concept, create new sets of surveys to expand on and evaluate companies of a different nature. Other thought experiments could include selecting a few other software companies ranging anywhere from smaller startups to more well established corporations. Evaluating each of these companies requires a shift in perspective to be able to see what the importance of software management is for making each environment successful in its own right. Management in a small startup may need emphasis in some areas more than others, as with a bigger company of 1000+ employees. As mentioned in this body of work, project management spans across many areas of the software development cycle, only four of which were evaluated and presented in this paper. Future work and implementation of this work in the real world would have to take other areas into consideration as well.

Moreover, validation of this work needs to be considered. The only way to take the metric from theory to a valid, functional metric would be to conduct long-term studies on actual software companies using the proposed methods; something which is out of the
scope and possibility of this master’s thesis project within the available resources and timeframe. Research programs would need to be developed to create surveys using the same approach, design and implement improvement plans based on the results of the administered surveys, and finally re-evaluate the quality of management after an appropriate length of time to validate the effectiveness of the methodology. Ideally, this would be done multiple times using the same sets of surveys on corresponding companies. The more companies the studies are conducted on, the more precisely validity can be assessed. The goal is for a statically significant set of results to show improvement in the quality of the software management using the proposed methodology.

It is important to be able to think about and work with software development practices to be able to start generating consistent methodologies. These methodologies can then be used as the foundation for practicing software engineering. The lack of such trusted methodologies is what leads to so many failed software projects. Software development should be as consistent as any other engineering field with as strict and regulated metrics and quality assessment practices. This paper is an addition to the resources available on the implementation of quality software practice - specifically in the area of management.
References


