University of Mississippi

eGrove

Electronic Theses and Dissertations | Graduate School

2019

Hitting the Bullseye: The Influence of Technical Debt on the Accuracy of Effort Estimation in Agile Projects

James Ball
University of Mississippi

Follow this and additional works at: https://egrove.olemiss.edu/etd

 Part of the Business Administration, Management, and Operations Commons

Recommended Citation

Ball, James, "Hitting the Bullseye: The Influence of Technical Debt on the Accuracy of Effort Estimation in Agile Projects" (2019). *Electronic Theses and Dissertations*. 1582.
https://egrove.olemiss.edu/etd/1582

This Dissertation is brought to you for free and open access by the Graduate School at eGrove. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of eGrove. For more information, please contact egrove@olemiss.edu.

Hitting the Bullseye: The Influence of Technical Debt on the Accuracy of Effort Estimation in Agile Projects

A Dissertation
Presented for the Doctor of Philosophy Degree
in the School of Business Administration
The University of Mississippi

by

JAMES F. BALL, JR.

May 2019

Copyright James F. Ball, Jr.
ALL RIGHTS RESERVED

ABSTRACT

As firms rapidly develop solutions in order to increase revenue and market share, software development decisions considered to be temporary shortcuts and/or compromises may be implemented. These shortcuts represent "technical debt," a metaphor which succinctly describes a software solution that should be "paid in full" or remediated in the future. Software architects and developers intend to resolve the "debt" in future product releases, but practitioners recognize that the challenge of always innovating may indefinitely postpone this remediation effort. Further, the accumulation of technical debt may have long term impact on the product's maintainability by the software development teams and, consequently, impact the effort estimate delivered to management for forecasting product delivery timelines and product revenue expectations. While there are multiple publications that have studied effort estimation in traditional and agile software development strategies, there is limited research which considers technical debt during the estimation effort. As a result, the purpose of this dissertation is to design and propose a research model intended to determine whether or not the consideration of technical debt during the effort estimation process will improve the accuracy of the effort estimate in an agile project.

DEDICATION

I dedicate this dissertation to my family, specifically my parents James and Dorothy Ball, my wife Beth Tutor Ball, my son Luke Thomas Ball and my daughter Rebekah Ruth Ball. I am forever grateful for the role each of you played in supporting my educational goals. Your lessons of perseverance, intelligence, humor and kindness made this journey possible and joyful. Thank you all very much!

ACKNOWLEDGEMENTS

I am grateful for the guidance and support of my dissertation committee, namely Dr. Brian Reithel, Dr. Anthony Ammeter, Dr. Bart Garner and Dr. Dawn Wilkins.

I am especially grateful to Dr. Brian Reithel for his persistent, positive guidance in the pursuit of this important research topic. Related, his support of my efforts to pursue a Doctorate late in my career was instrumental in preparing me for the academic journey while working full-time.

I want to thank Dr. Ammeter for his thoughtful contributions especially in the areas of my dissertation involving team research. His willingness to meet and patiently explain key aspects of team theory as well as the pitfalls was very beneficial.

Dr. Wilkins provided me with valuable insights especially involving the extensions of this dissertation. That is, by utilizing machine learning techniques, it may be possible to improve the detection of technical debt and its direct impact on the effort estimation process.

Dr. Garner shared a common understanding of research combined with the practitioner's mindset. Having a committee member with a combination of practitioner and theoretical focus was very helpful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I
INTRODUCTION

How long will it take? This fundamental question is routinely posed to software development practitioners in an attempt to understand the potential delivery date and corresponding budget of a project or effort. The motivation for the question may serve a number of purposes. For a deadline-driven manager, the inquiry could signal a desire to accelerate the delivery of a software solution. For the corporate accountant, the request could correspond to a request for a time component variable in a more complex financial equation aimed at giving executive management a budget forecast. Regardless of the initial motivation for the question, the unspoken desire is accuracy. Of course, early delivery would be considered the most desirable outcome, but being able to consistently and accurately forecast the delivery date is beneficial in any work environment. In a software development environment and throughout this dissertation, this desired outcome is called an *accurate effort estimate*.

1.1. Background.

With a desire for accuracy in our effort estimates, extending the forecasted deadlines is an obvious concern for the software development practitioner. There are a number of studies that attempt to address the inaccuracies of effort estimation. One of the first known papers addressing software effort estimation was published by Farr and Nanus in 1964 (Farr and Nanus 1964). In their publication, Farr and Nanus succinctly summarized the problem by saying “estimates have

historically been very unreliable." In an attempt to address the estimation difficulties, they identified approximately fifty factors that influenced the cost estimate effort of a software project and focused primarily on the cost factors.

Fast forward thirty years and we see the software industry was still faced with effort estimation challenges. As a part of their Project Smart, The Standish Group International published results that validated the specific concerns of inaccurate effort estimations (Clancy 1995). In 1995, they surveyed IT executive managers and determined that for projects that reached completion, the time estimate average was 222% of the original time estimate. Related, the survey reported on cost overruns, where the budget overrun averaged 189% of the original forecasted budget. Further, as cited by Laqrichi, Gourc, and Marmier (Laqrichi et al. 2015), a more recent study conducted by The Standish Group International found that 44% of software projects extended beyond the expected costs and time estimates.

Giving further evidence, the Harvard Business Review (HBR) claimed to have conducted one of the largest global studies of IT change initiatives (Flyvbjerg and Budzier 2011), where change involves new product development as well as product enhancement. After examining 1,471 projects, their findings showed a black swan or surprising finding. Specifically, their research revealed that one out of every six projects incurred a cost overrun of 200%, on average, and a schedule overrun of almost 70%. HBR noted that by only looking at the overall averages, consultants and IT managers are missing these damaging outliers that can dramatically impact corporate budgets and negatively impact the careers of technology executives.

Finally, a recent survey conducted by Coding Sans evaluated the current state of the software development industry from the perspective of software firms. Among other findings, their survey revealed that effort estimation is ranked as the second highest cause of software

developer problems ("Software Development Trends 2018" n.d.). Their Figure 1.1 below shows the top ten reasons for delivery problems.

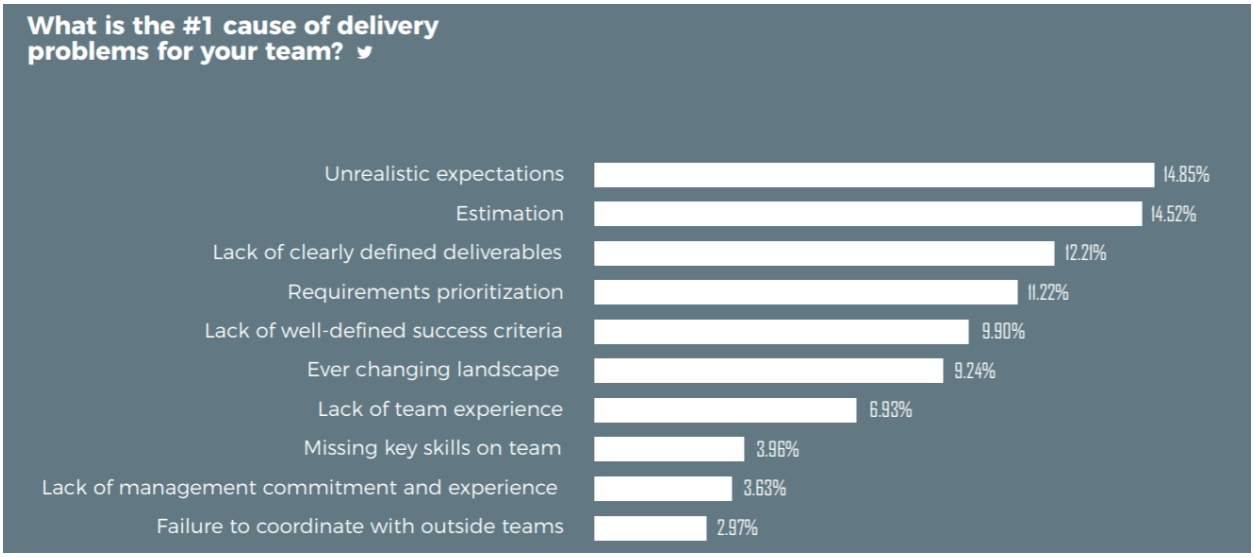


Figure 1.1. Coding Sans' visualization of the reasons for delivery problems ("Software Development Trends 2018").

While the challenges of consistently achieving accurate effort estimates have persisted for as long as software development has been a professional field, the software development community has evolved in an attempt to improve a variety of factors of software development including effort accuracy. In the early 1980s, Boehm published a current state of the software engineering landscape and highlighted a number of the more popular cost estimation techniques while promoting his Constructive Cost Model (COCOMO) technique (Boehm 1981). In his Figure 1.2 below, Boehm visually explains his perspective on the challenge of accurate effort estimates. In short, Boehm suggested the accuracy of the estimate improved as the software development project moved, from left to right, through the various phases until the software is ready for delivery to the customer. This graph is commonly referred to as the Cone of Uncertainty.

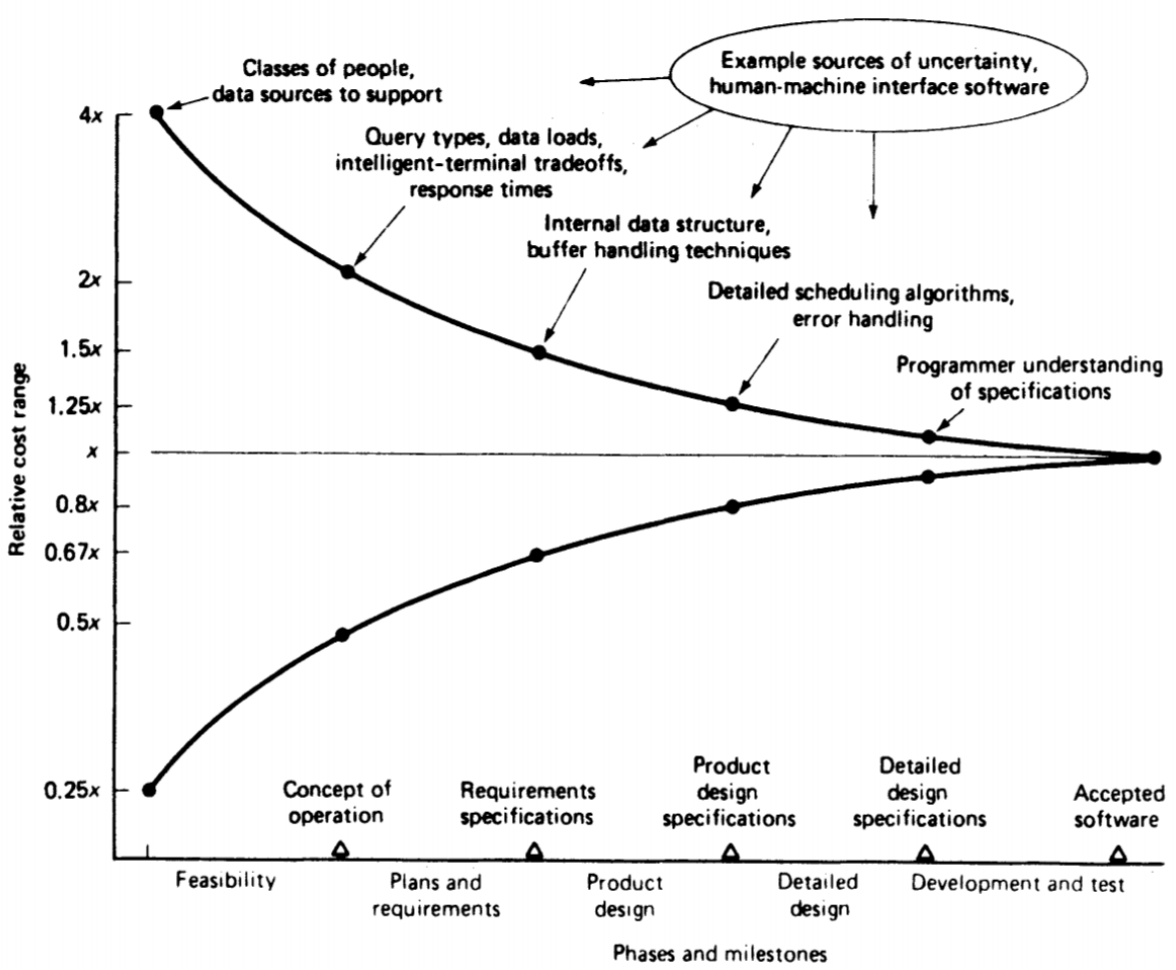


Figure 1.2. Boehm's visualization of effort accuracy during each development phase (Boehm 1981).

While the traditional techniques presented by Boehm are still in practice today, an alternative software development model has gained acceptance, namely *agile*. Agile software development principles brought the promise of improved quality and speed of product delivery ("Manifesto for Agile Software Development" 2001). When introduced, agile touted a customer-first strategy, where development engaged in frequent communication with business professionals and continuously developed leading to more frequent product releases. As it relates to the accuracy of effort estimation in a software development project, the Cone of Uncertainty comes under fire. Specifically, the agile development model presents the notion that as time marches on, the accuracy of the estimate will suffer. In a nutshell, the agile philosophy is

built upon the assumption that the details of customer requirements will change over time. In some ways, a 180-degree rotation of Figure 1.1 may depict the agile perspective more accurately. That is, as the time before the customer can evaluate the software increases, the accuracy of meeting the customer expectations may decrease.

In parallel with the popularity of agile software development lifecycle efforts with their rapid release cycles, software development teams may begin to recognize the impact of poor historical code design on future code enhancements. These poor designs represent *technical debt*, a metaphor which succinctly describes a software solution that should be "paid in full" or remediated in the future (Cunningham 1992). Software development teams aware of this debt typically intend to resolve the debt in future product releases, but practitioners recognize the challenge presented by always innovating may indefinitely postpone this remediation effort. Furthermore, the accumulation of technical debt may have long term impact on the product's maintainability by the software development teams and, consequently, impact the effort estimate delivered to management for forecasting product delivery timelines, product revenue expectations, and software development budget forecasts. However, even when this debilitating debt is recognized, it is unclear if or how that debt is considered during a software development team's effort estimation exercises.

1.3. Problem Statement.

This study has a primary contribution. The purpose is to design and propose a research model intended to determine whether or not the consideration of technical debt during the effort estimation process will improve the accuracy of the effort estimate in an agile project.

A review of previous studies uncovered some key findings of prior work:

1. Agile software development techniques are being studied; researchers continue to indicate that more research is required.
2. Researchers have studied technical debt prevention techniques as well as technical debt detection. However, the studies are tied almost exclusively to agile software programming efforts without consideration of any human or team factors.
3. As noted in prior sections, the accuracy of effort estimates in software development projects remains a top priority for the software industry.
4. After a thorough review, a gap in the research has been identified as it relates to technical debt. Specifically, there appears to be no research model that incorporates technical debt into an existing effort estimation process involving agile software development projects.

Researchers recognize the challenges of determining "how long it takes" to deliver a software product and have provided various software development methodologies and techniques to address inaccurate effort estimates in software development without reaching a definitive solution. Furthermore, corporations desire accurate effort estimations for the timely delivery of products and accurate forecasting of revenue potential as well as budget forecasts. However, prior studies indicate that over fifty years of software development maturation have not resolved the challenge of inaccurate effort estimations for software development projects. There remains a need to provide further clarity on the factors that influence the accuracy of an effort estimate.

For that purpose, this dissertation intends to provide insights into this research question: "*In an agile software project, does the consideration of technical debt during the effort estimation process improve the accuracy of the effort estimate?"*

1.4. Problem Significance.

The significance of this problem is evident to researchers and software development practitioners alike. Even with the adoption of modern software development lifecycles and strategies, prior research shows the accuracy of effort estimates remains a challenge affecting the revenue, budgets, and careers of technology professionals worldwide. Research that advances a solution to the challenge of an accurate effort estimate is well overdue.

1.5. Defining Technical Debt.

As a key factor of this dissertation's research model, it is important to understand technical debt in a software development context. Since 1992, when Ward Cunningham associated the term "debt" with software development in his report on the WyCash Portfolio Management System (Cunningham 1992), technical teams have clutched on to the term to characterize the problems they face when developing and maintaining software applications.

Today, technical debt is considered a computer programming metaphor where software developers take real or perceived implementation shortcuts during initial software development that eventually must be "paid" in the future. The "payment" is typically in the form of rewriting components of the software application. Until the "payment" is made in full, the impact could be any number of work-a-round or manual activities, including developing applications to monitor for errors, hiring additional employees to manually perform duties, and even limiting the application functionality for customers.

At times, explaining a software development concept outside the software world helps explain a concept. Consider a Chief Information Officer (COO) for a major automobile manufacturer being challenged by the board to improve the production capabilities in

manufacturing facilities. The COO assembles his management team and succinctly delivers the edict for a new focus. After all, time is money. After reviewing the manufacturing steps and customer satisfaction surveys, the management team determines that windshield wipers involve a manual installation process. After analyzing the available customer data on the criticism and need for windshield wipers worldwide, the management team recommends shipping thousands of new cars without windshield wipers. The idea is approved and implemented. Now, as new owners purchase these wiper-less cars and experience inclement weather, these same owners begin to call upon or visit dealerships to complain about the lack of wipers. As the bad news travels to corporate headquarters, the COO's management team recognizes the problem, but ignores it for now and directs complaints to their customer service department staffed with individuals who attempt to reason with the owners. But as new car owners experience more bad weather as well as windshield insect splatter, complaints rise and new car sales for the automobile manufacturer begin to plummet. It now becomes evident to the COO that windshield wipers are needed. So, the COO orders a mass recall to install wipers.

This fictitious example presents the case for the core issue of technical debt that concerns practitioners. In this example, the cost of installing wipers at the manufacturing plant would have been less expensive than the recall costs necessary to notify the impacted customers, to schedule visits at local dealerships, to ship wipers to the dealers, and to perform the same wiper installation manually, possibly paying overtime wages to repair specialists interested in resolving customer complaints as soon as possible. The cost of performing a task properly initially is less than the task remediation costs in the future.

Returning to agile software development, a modern concept of technical debt was addressed by Kruchten, Nord, Ozkaya, and Falessi. The authors noted that the metaphor originally introduced by Ward Cunningham in 1992 may still be confusing to some in the current industry (Kruchten et al. 2013a). They provide a more recent definition, articulated by Steve McConnell. In his words, technical debt is a consequence of "a *design or construction approach that's expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now (including increased cost over* time)." Today, technical debt is most commonly associated with agile programming projects.

According to a survey conducted by Lim, Taksande, and Seaman against thirty-five practitioners from British Columbia and the United States, there is evidence that technical debt exists (Lim et al. 2012). Their findings indicated that while seventy-five percent of the respondents did not recognize the term technical debt initially, upon giving a brief definition of the metaphor, all practitioners except two understood the concept immediately.

What causes technical debt? There is evidence of multiple causes. Lim et al. (Lim et al. 2012) reported that practitioners from their study did not see sloppy programming or poor developer discipline as the cause, but rather the "intentional decisions to trade off competing concerns during development." Similarly, Kruchten, Nord, and Ozkaya (Kruchten et al. 2012) suggest that most authors point to schedule pressure as the major cause of technical debt. However, they identified other potential causes including human factors such as carelessness, lack of education, and incompetence as well as non-human factors such as poor processes and a lack of automated quality control (Kruchten et al. 2012). While these prior research efforts suggest schedule failure risks are a major cause of technical debt, there has been no prior study that evaluates the impact of schedule uncertainty on the perceived level of technical debt.

1.6. Conceptual Model.

Figure 1.3 provides a visual of the conceptual model of this study, referred to as the "Factors Influencing Agile Development Effort Estimation Accuracy" (FIADEEA) model.

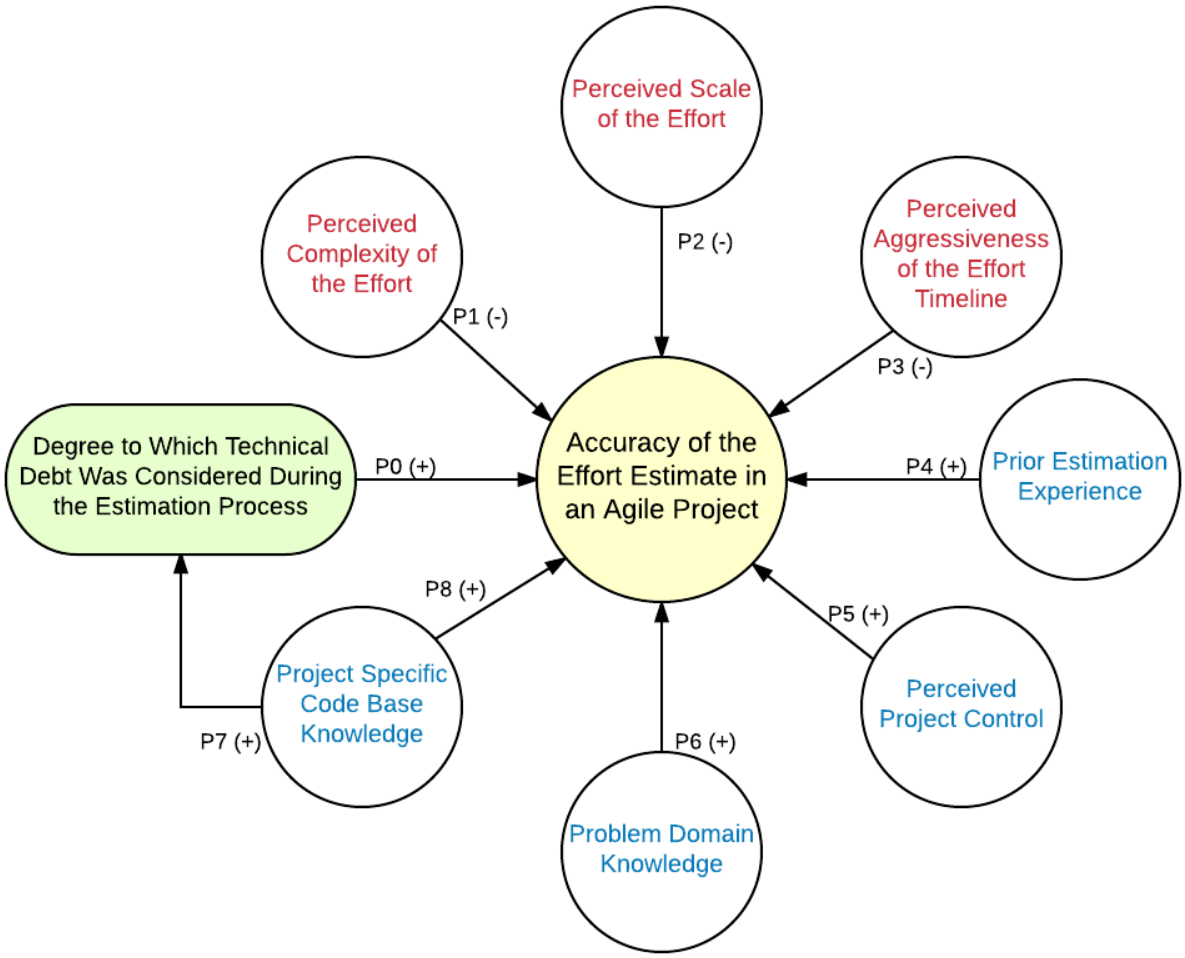


Figure 1.3. A visual representation of the FIADEEA model.

In the center of the model is the dependent variable, the Accuracy of the Effort Estimate in an Agile Project (AEE). To the left of the FIADEEA model is the key construct for this study, the degree to which technical debt was considered during the effort estimation process. We also attempt to group the factors above and below AEE. Along the top of the model are the project-

based factors. Along the bottom of the model are the team-based factors or factors that are evaluated from the perspective of the team instead of individuals.

Key definitions of the conceptual model are provided below while related propositions will be provided in line with the corresponding definitions.

Accuracy of the Effort Estimate in an Agile Project (AEE). AEE is the degree to which the estimated timeline of the effort concurs with the actual effort. By measuring the difference between the estimated effort time and the actual effort time, the effort estimation accuracy can be determined for an agile project.

Degree to Which Technical Debt was Considered During the Estimation Process (TD). The degree to which agile team members considered technical debt in the existing source code, libraries or modules during the effort estimation task. And while agile teams may recognize the influence of technical debt on project delivery, prior to this study, technical debt has no clear presence in popular agile effort estimation techniques.

> **P0:** *The Degree to Which Technical Debt was Considered During the Estimation Process (TD) is positively related to AEE.* This proposition is intended to be one of the key contributions of this dissertation. Prior research has not included TD in estimation models or efforts. This dissertation postulates that the consideration of technical debt (TD) will improve the effort estimate in an agile project.

Perceived Complexity of the Effort. The extent to which the number of components and associated component interactions is considered challenging. Prior research by Ziauddin et al. (Ziauddin and Zia 2012) emphasized the importance of the complexity impact on the agile effort estimation. They suggested the use of a User Story Complexity Scale unique to the agile team,

which would rank the effort based on a number of factors including research requirements, difficulty of judgement calls, system or subsystem dependencies as well as the need for refactoring.

> **P1:** *The Perceived Complexity of the Effort is negatively related to AEE*. Consistent with prior research, this dissertation also suggests that effort complexity affects agile projects. As complexity increases, the inability for agile team members to "see" the impact of the component interactions will negatively influence the ability to accurately estimate the effort.

Perceived Scale of the Effort. The size of the effort. Perceived scale is concerned with factors that influence the size of the effort including the projected budget costs, personnel count, geographic involvement, and project duration, just to name a few. An example of a large-scale project would be the replacement of an accounting system at a Fortune 500 corporation.

> **P2:** *The Perceived Scale of the Effort is negatively related to AEE.* As mentioned in the definitions section, scale is the size of the effort. While some researchers may combine scale with complexity, this dissertation intentionally segregates the two constructs in order to capture the influence of project size.

Perceived Aggressiveness of the Effort Timeline. The degree to which a project's timeline is compressed below the ideal timeline.

> **P3:** *The Perceived Aggressiveness of the Effort Timeline is negatively related to AEE.* For individuals with a software development background as a practitioner, it "makes sense" that developers would argue that an increase in aggressive timelines would reduce AEE. This dissertation intends to include perceived aggressiveness as a

construct to determine its influence on AEE from the perspective of developers, technical project leaders (e.g., scrum master) and other agile project team members.

Prior Estimation Experience. The years of experience involved with estimating effort in software development projects. According to Usman et al, not all agile team members use effort estimation.(Usman et al. 2014).

P4: *Prior Estimation Experience is positively related to AEE.* This dissertation expects seasoned software developers as well as seasoned software development teams to have an advantage on knowing how to perform software effort estimations and should give more accurate estimates.

Perceived Project Control. The degree to which the effort timeline can be influenced. The belief by the agile team of having some influence on the project deliverables. For example, an agile team may desire to postpone new project enhancements brought forward by a client until a current agile iteration known as a "sprint" is complete. Having project control means the software development team can influence the internal or external customer to postpone the enhancement request and avoid jeopardizing the delivery.

P5: *Perceived Project Control is positively related to AEE.* An increase in project control gives the agile team members increased confidence that their project timeline estimates will not be negatively affected due to changing priorities or requirements.

Problem Domain Knowledge. The degree to which agile team members are familiar with the discipline or field that the application encompasses. Throughout an agile project, agile team members will plan and develop solutions to address various programming problems. A

combination of the agile team's skills, experience and familiarity (a.k.a., program domain knowledge) with each type of problem will guide the team towards specific solutions.

> **P6:** *Problem Domain Knowledge is positively related to AEE*. When the agile team or team members have more familiarity with the business or technical problem which the software development effort intends to address, this dissertation assumes their ability to make accurate effort estimates increases.

Project Specific Code Base Knowledge. The degree to which the agile team members have past experience with existing source code, libraries or modules that will be utilized in the software development effort.

> **P7:** *Project Specific Code Base Knowledge is positively related to the degree to which TD was considered during the effort estimation process*. When an agile team has familiarity with significant portions of a particular code base, this dissertation expects the team's visibility into areas of potential technical debt to be high.
>
> **P8:** *Project Specific Code Base Knowledge is positively related to AEE*. This dissertation postulates that increases in code base knowledge increases the ability to accurately estimate the effort.

1.7. Proposed Methodology.

This dissertation and corresponding study will follow the guidance of Churchill (Churchill 1979). This includes instrument development and conducting a pilot study to purify the instrument. Afterwards, a main study will be conducted followed by assessing the reliability and validity, and finally, developing empirical conclusions.

Practitioners having experience with agile software development methods are the target audience of both the pilot and main studies. Computer programming skills are not required to answer instrument questions. This allows for participation from practitioners from various agile areas including project management and quality control. The entire survey instrument can be found in APPENDIX A.

To test the validity of the instrument, factor analysis will be utilized during the pilot study provided there is sufficient data. If not, Exploratory Factor Analysis (EFA) will be used during the main study. In short, EFA does not assume any a-priori relationship with the constructs and, as a result, should aide in the validation of the survey instrument (Petscher et al. 2013). Reliability testing will involve Cronbach's alpha measurements (Jarvis et al. 2003). Consistent with an acceptable level of measured reliability in academic research, the alpha coefficients should exceed 0.7.

3.3.1. Conducting the Pilot Study.

A pilot study will be conducted at two nearby software firms where agile software methods are utilized on a daily basis during their software development life cycle (SDLC). The participants will be part-time or full-time practitioners with prior agile experience.

After the refinement of the instrument, the main study will be conducted with the revised instrument. Beyond basic analysis using descriptive statistics, regression will be used to perform data analysis on the main study (McCullagh 1980).

1.8. Chapter Overview.

The purpose of Chapter I was to share some background concerning the complex challenge of accurately providing an effort estimate in an agile software development project, to

introduce technical debt as a key factor in agile software development, and to propose a plan for researching the influence of considering technical debt during the effort estimation process of an agile software development project.

The remaining chapters in this dissertation will be as follows. Chapter II will review the related literature covering each of the key components in the conceptual model, graphically depicted in Figure 1.2, namely effort estimation, technical debt, agile software methodologies, team development and project management. Chapter III will introduce the conceptual model and propositions in greater detail while also presenting the methodology for both the pilot and the main studies. Chapter IV will provide details of the pilot study effort and results, as well as the main study and results, and a discussion of those results. Chapter V will cover the empirical conclusions, discuss the limiting conditions of the study and discuss implications for researchers and practitioners. Finally, Chapter VI will conclude the study with some discussion of future research directions.

CHAPTER II
LITERATURE REVIEW

Before describing the conceptual model for factors that impact the accuracy of the effort estimate in an agile project, an examination of previous published research is needed. The focus of this literature review is on five areas of study: effort estimation, technical debt, agile software development projects, project or project management factors, and team development factors. The intent is not to exhaustively cover all aspects of these topics of study, but rather to focus on the key research areas related to this topic of study and to uncover any potential opportunities that warrant further study. First, the key factors that impact effort estimation are discussed, followed by a review of related effort estimation studies.

2.1. Technical Debt.

As a first step in exploring technical debt, it is necessary to move from the vague metaphor established by Ward Cunningham (Cunningham 1992) and recognize a more current concept of technical debt in a software development environment. A modern definition of technical debt is presented by Kruchten, Nord, Ozkaya, and Falessi. The authors noted that the metaphor originally introduced by Cunningham may still be confusing to some in the current industry (Kruchten et al. 2012). They provide a more recent definition, articulated by Steve McConnell. In his words, technical debt is consequence of "a *design or construction approach*

that's expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now (including increased cost over time)."

With this modern definition in mind, it is important to note that technical debt is the consequence of a decision. Further, technical debt should not be equated to software defects or software bugs. While previous expedient decisions in a technical context may introduce bugs, even software developed without expedient decisions includes functionality that, under new or uncertain circumstances, will generate incorrect results or cause unexpected errors. Kruchten et al. provided a simple chart to distinguish technical debt from defects/bugs, shown in Figure 2.1. This categorization shows the perspective of the client or customer, noting that defects and technical debt offer negative value but they are different by their visibility to the customer. While it isn't safe to assume that all technical debt is invisible to the customer, the chart serves the purpose of distinguishing technical debt from software bugs or defects.

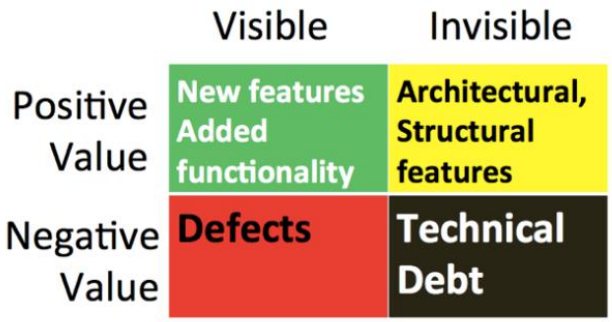


Figure 2.1. Categories of backlog items to be developed by an Agile team (Kruchten et al. 2012).

Of particular interest to this study is the notion that the "cost" of delaying an appropriate design or construction approach increases over time. Besides the potential financial impact of these expedient design or construct approaches, this dissertation intends to evaluate the consideration of technical debt during the effort estimation task in an agile software development environment.

With technical debt defined, attention can now be turned to the actual literature review of technical debt. In the past fifteen years, most of the research attention towards technical debt considered only the technical factors. Numerous publications were found to have addressed the technical aspects of identifying, preventing, and reducing technical debt in a code base. Three important publications were identified that discussed the types of technical debt in a software development environment in detail.

Alves, Mendes, de Mendonça, Spínola & Shull performed a comprehensive literature review aimed at mapping strategies intended to identify and manage technical debt (Alves et al. 2016). Their specific research question was, "What are the strategies that have been proposed to identify or manage technical debt in software projects?" Figure 2.2 shows their categorization of technical debt research. A key finding noted by the authors was the absence of real-world studies in the software industry.

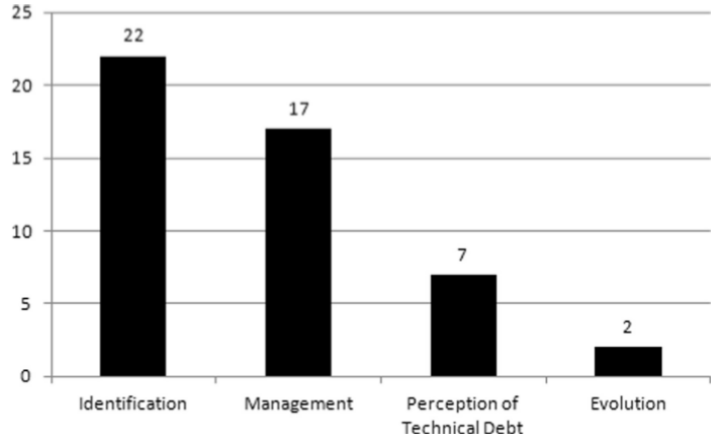


Figure 2.2. Alves et al. categorization of technical debt research (Alves et al. 2016).

At Google, Morgenthaler, Gridnev, Sauciuc, & Bhansali provided an in-depth discussion covering the technical strategies for dealing with various types of technical debt in existing code bases (Morgenthaler et al. 2012) across domains. The authors referred to this as build debt. While it is not necessary to cover their categorization of technical debt, the authors

acknowledged technical debt as a problem that must be addressed proactively across all Google software projects. Their specific advice for Google developers in dealing with debt can be summarized with these three principles implemented at Google.

1. Use automation. Google encourages the use of automated tools to detect and even remediate technical debt found across Google software projects and teams.
2. Make it easy to do the right thing. Morgenthaler et al. recognized that developers may introduce technical debt because they are unaware. As a result, there was interest in detecting and preventing these early causes of technical debt during code editing or code review. The authors noticed this was a non-trivial problem.
3. Make it hard to do the wrong thing. The authors noted examples that would include compile-time and link-time decisions that would prevent Google developers from creating a dependency on software that is not ready for deployment and to build more strict checks that generate compile time errors and warnings.

In conclusion, Morgenthaler, et al. encouraged organizations to "pay attention to the debts early" and create tools and an environment that encourages the continuous repayment of technical debt.

Separately, Kazman, Cai, Mo, Feng, Xiao et al. focused on architecture debt, a type of technical debt that is invisible to the customer but impacts to ability to enhance the product over time. In short, the authors' primary research question was, "Are the identified architecture issues the root cause of technical debt?" Using their tool, Titan (Xiao et al. 2014), to analyze the software source code architecture, the authors performed a case study with one software company and had promising results to support their hypothesis. A key finding involved their calculation on a return of investment (ROI) of at least thirty percent by refactoring the architecture to attempt to reduce or eliminate the technical debt (Kazman et al. 2015).

Aside from the articles that focused on the technical factors involving the prevention and detection of technical debt in software development, relatively few studies were identified with a focus on the human factors that influence technical debt. Acuna, Juristo, & Moreno proposed a framework for assigning developers to the correct roles in a software development team, based on their personality traits (Acuna et al. 2006). They used Raymond Cattell's 16PF personality questionnaire as a starting point (Cattell et al. 1970). The authors' recommended using their role assignment findings as a tool for rational guidance to assist managers in role assignment. Their findings were not intended to be a final determining factor of role assignment of developers. Instead, their study gave software managers another tool to evaluate appropriate roles in conjunction with preferences, job levels and technical knowledge.

Avgeriou, Kruchten, Nord, Ozkaya, & Seaman covered a broad range of topics associated with technical debt, including a perspective that technical debt begins at the outset of the software project (Avgeriou et al. 2016). Most notably, four triggers of technical debt were suggested, namely: process, management, context and business goals. The authors emphasized the importance of researchers, educators, practitioners, and tool developers needing to work together to address the ongoing concerns of technical debt.

Fortunately, or unfortunately, no prior research combining technical debt with effort estimation was found. This gap in the research gives hope that this dissertation will make a useful contribution to the body of work dedicated to exploring the management of technical debt in agile software development efforts.

2.2. Agile Software Development.

Central to our study of effort estimation is a narrowing of the focus to one software development lifecycle model, namely agile software development. Agile software development principles brought the promise of a customer-first mindset, frequent communication between developers and business professionals, and continuous development leading to more frequent product releases, all in an effort to improve the quality and speed associated with the delivery of software products. As discussed by Ruparelia (Ruparelia 2010), agile software development is one of several software development lifecycle models available to software teams. Specific branches of agile programming include crystal, extreme programming (XP), joint application development (JAD), Lean Development (LD), and Scrum. In 2001, a group of software developers met to discuss what is common among the different agile branches and out of their meetings came the Agile Manifesto. ("Manifesto for Agile Software Development" 2001). While the manifesto efforts may not have contributed any particular changes to agile development branches, it is widely considered to be a major contributor to a turning point affecting the growing popularity of agile software development among practitioners.

Currently, there is evidence that agile techniques are in vogue. A recent industry survey conducted by HP Enterprise showed the overall growth pattern of agile (*Agile Is the New Normal* 2017). Interviewing 601 development and Information Technology (IT) professionals, their survey indicated an annual increase of reported agile adoption. Figure 2.3 shows their growth chart of agile adoption over a ten-year period. These results give evidence of the importance of agile in the software development industry and warrants emphasis as a key topic of this research's literature focus.

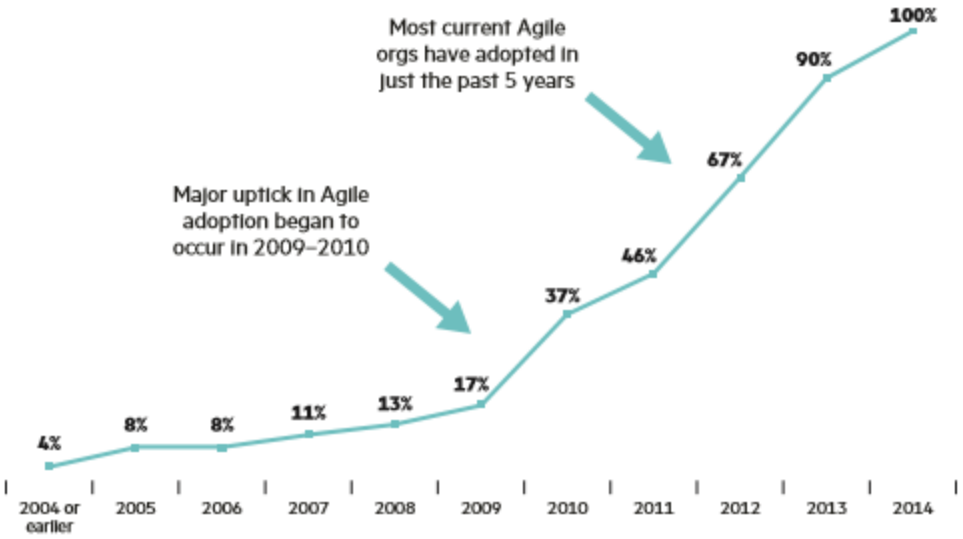


Figure 2.3. Ten-year Agile adoption rate (*Agile Is the New Normal* 2017).

Dybå and Dingsøyr provided a thorough, systematic review of research literature focused on agile software development (Dybå and Dingsøyr 2008). They identified 1,996 studies in all, including 36 empirical studies. As a part of their research review, they categorized the empirical studies into four major categories: (1) introduction and adoption, (2) human and social factors, (3) perceptions of agile methods and (4) comparative studies. Based on their analysis, the majority of the empirical studies focused on the agile method known as extreme programming (XP). From their review of the empirical studies, the authors noted that agile methods in these studies were considered to contribute to an increase in productivity as well as quality control when compared to alternative SDLC techniques. However, the authors questioned some of the study participant recruitment strategies that would ensure an unbiased comparison. Finally, the authors noted that future research focus on Scrum, a growing agile branch popular among practitioners, warranted attention. It is worth noting that neither effort estimation nor technical debt were discussed in their thorough review.

A more recent study discussed the emerging themes of agile software development (Dingsøyr and Lassenius 2016). Their Figure 2.4 graphically shows the relative count of publications focused on agile software development over a ten-year period. Their search and categorization efforts were taken from a single search source, the Scopus scientific database. The authors noted the decline in research interests involving XP agile methods along with a significant emphasis on the Scrum techniques. At the time of their study, the authors noted that practitioners may prefer the continuous integration agile methods, but they provided no empirical evidence for their assertion. Continuous integration promotes the continuous and automated verification of components prior to integration. While the authors may consider "Continuous Integration" an agile technique, the act of increasing the frequency and automated testing and integration of new features into the main or master product multiple times per day may be incorporated into any software development life cycle technique.

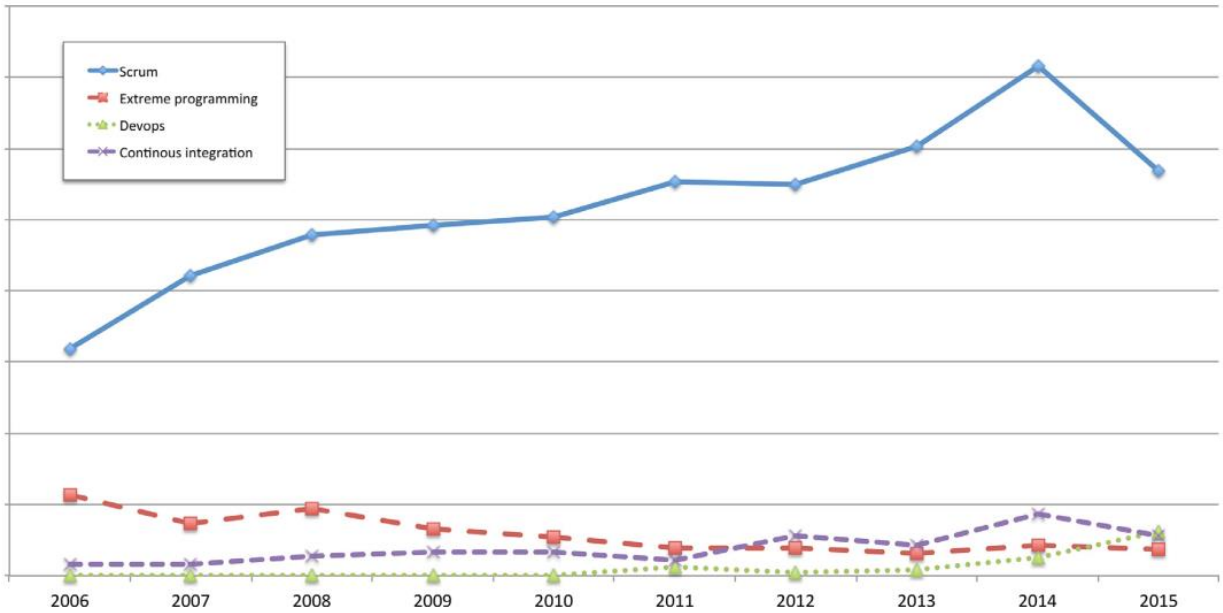


Figure 2.4. Relative interest over time in various Agile techniques (Dingsøyr and Lassenius 2016).

Of course, agile software development methods are not without warts. Turk, France and Rump covered the underlying assumptions of the Agile Manifesto and how the assumptions may introduce limitations for the particular software development project (Turk et al. 2014). In the authors' words, "Agile approaches are not process silver bullets." Because agile methods emphasize frequent team communication and speed of delivery, there are two limitations that agile practitioners routinely have to address. First, team complexity can be created by geographical dispersion of the software teams, including subcontracted teams as well as large development teams. Core to the principles of agile software development is the ease of communication among the teams. However, with the team complexity heightened, as outlined above, communication may be non-trivial and possibly prohibitive, especially in the case of geographic dispersion. Second, the authors pointed to product complexity as the other key factor that may challenge the assumptions of agile software development. Besides product complexity involving multiple intricate components, safety critical software may prohibit rapid release cycles and require a more formal testing process, creating a hybrid software development model that incorporates agile methods as well as traditional techniques that emphasize formal processes and testing to ensure safety. This added requirement will necessitate rigorous processes to maintain quality confidence.

As one potential solution to the challenges noted by Turk et al., Qurashi and Qureshi offer a solution for large scale teams called a "Scrum of Scrums" (Qurashi and Qureshi 2014). Their Scrum of Scrums allows for multiple Scrum teams with the Scrum master or Scrum leader also participating as a Scrum member in a larger Scrum team. A key to success is to form the Scrum teams to address specific components or modules that allow for rapid communications

without the limitations that can be associated with cross-team development efforts. As pointed out by the authors, this layering can be extended to form a Scrum of Scrums of Scrums.

So far, the literature reviews addressing agile methods have not emphasized effort estimation or recognized technical debt. Literature that incorporates effort estimation with agile methods will be covered in section 2.5.

2.3. Project Factors.

Project factors that impact effort estimation are well-documented in the project management research literature. Rather than rehashing all possible project factors, this literature review focuses on key publications that are related to agile software development or effort estimation.

Dikert, Paasivaara, and Lassenius performed a literature review of large-scale agile software development efforts over a ten year period (Dikert et al. 2016). The authors quickly pointed out the popularity of agile methods as applied to small or individual teams. However, what has been studied pertaining to the use of agile methods in large-scale transformations? Searching through online databases including ACM, IEEExplore, Scopus, and Web of knowledge, the authors found fifty-two related publications across forty-two industries, with forty-six publications considered to be experience reports without a research focus. Their research questions can be succinctly restated as follows.

1. "What challenges have been reported for large-scale transformations?"
2. "What success factors have been reported for large-scale transformations?"

The summary results of their study found the median size of the participating organization was 300 people with a wide range of 50 people to 18,000 people. Agile efforts

were also divided into teams, ranging from 6 teams to 150 teams. The authors found that different business industry areas appear to be represented evenly. As for the agile strategies chosen, Scrum was the most prevalent followed by Extreme Programming. The authors found the organizations often combined components of different agile methods including Scrum, Extreme Programming and Lean.

As for the challenges that prohibited the success in large-scale agile efforts, their research found nine categories of challenges: integration of non-development functions, difficulties in implementing agile, resistance to change, requirements engineering challenges, coordination challenges in a multi-team environment, lack of time and financial investment, hierarchical management and organizational boundaries, quality assurance challenges, and different agile approaches in a multi-team environment. Turning to the success factors in large-scale agile projects, the authors found eleven categories of successes. The combined list of team and individual success factors include management support, commitment to change, change leaders, customizing the agile approach, piloting, training and coaching, engage everyone, communication and transparency, mindset alignment, team autonomy, and requirements management. The authors concluded that the research is "seriously lagging behind" in this area of software development, specifically noting that the identification of challenge and success factors was largely based on practitioner perceptions, with no known relationship between the potential factors.

Vijayasarathy and Butler conducted a survey to evaluate organizational, project and team factors that impact the selection of the software development module (Vijayasarathy and Butler 2016). While their findings did not reveal any particular details of project related factors that impacted effort estimation, their study revealed a couple of key points. First, organizational size

was not a significant factor in determining the use of agile methods or not. Second, related to project factors, the use of agile methodologies was associated with the allocated project budget. Their study shows 50% of agile projects had a budget under $200K and 45.8% of hybrid agile efforts (agile combined with hybrid) had a budget between $200K and $1M. No other project factors were presented in their findings.

While it is appropriate to consider the project factors affecting effort estimation during a major rollout or transformation, it is just as important to consider the project factors post-release or during the maintenance phase of software development. This is commonly referred to as software maintenance. Banker, Davis, and Slaughter conducted a field study to evaluate how project decisions made during a rollout impact software maintenance efforts post-rollout (Banker et al. 1998). First, they examined situations in which the project software teams used code generators or software applications that translated business logic to actual source code in a particular programming language, such as Java or C#. Their field study revealed that a 10% increase in code generator use resulted in a 3.8% increase in software maintenance hours. In short, the maintenance developers found that a mixture of generated code with custom code made for difficult work in adding enhancements, increased compilation time and complicated debugging as well as testing. Second, the use of package software or third-party purchased software resulted in a 6.5% decrease in software enhancement project hours. No specific conclusions were provided, but the authors noted that software managers and maintenance programmers were surprised by this finding.

2.4. Team Development Factors.

Similar to project factors, other factors such as team, team development and team size factors may be considered a traditional factor impacting software development projects and

effort estimation techniques. Revisiting the study conducted by Vijayasarathy and Butler, their findings supported a relationship between the selection of agile software development strategies and the number of teams. See Figure 2.5.

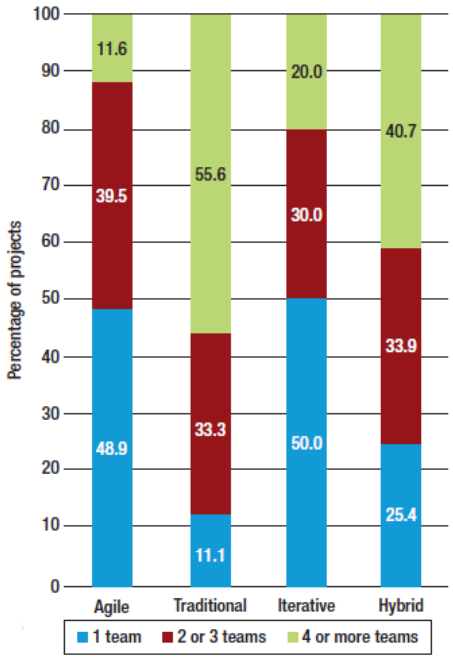


Figure 2.5. Relationship between software methods and number of teams (Vijayasarathy and Butler 2016).

Absent from Figure 2.5 is the relationship of software methodologies chosen based on the team sizes. In short, the authors also determined that agile was the preferred method for teams of ten or fewer. While their study is not of particular interest to a study in effort estimation, it reveals that the adoption of agile methods is preferred when one to three teams are engaged in the effort while traditional software methods are preferred when there are four or more teams.

Espinosa, Slaughter, Kraut, and Herbsleb evaluated the relationship of task and team familiarity with complexity in a geographically distributed software development environment. While the notion that task familiarity (i.e., similar or past task experience) and team familiarity (i.e., previous work experience with teams and team members) benefits effort performance, the

authors uncovered enlightening findings as the task complexity increased. Consistent with practitioner intuition, the authors determined that familiarity helped offset the negative influence of the task or project size. However, their empirical evidence suggested that familiarity may hurt performance in more complex tasks. In fact, their findings reveal that less familiar team members are able to "innovate and attain higher levels of performance with complex tasks."

Whitworth and Biddle conducted a qualitative study into the social process that contribute to the success of agile teams (Whitworth and Biddle 2007). Their findings showed the benefits of the daily stand-up meeting, where developers reveal their activities and near-future plans to the agile team. This "what have you done for me lately" mindset provided some peer pressure for developers to show daily accountability and progress in individually solving problems and revealing complications in solving a problem. Simultaneously, the developers on an agile team did not feel these daily sessions to be negative. The sessions provided awareness and transparency into the efforts of all team members. Their study also revealed that "information radiators" or visual representations of team progress served as sources of motivation, excitement and team cohesion. As for pitfalls, the authors' study revealed the stress created for some developers of the "always communicating" culture among some agile teams. Also, while these agile teams were increasing in their internal transparency, there was a strong inclination to focus on developer activities only, which may negatively impact quality assurance, business analysts, and technical writing roles, along with their deliverables. The authors noted further study into diverse configurations of agile practices were warranted to solidify their initial findings.

Overall, the literature review of team development factors indicated a potential influence of teams and team development on performance, but no specific study was found that combined team research and effort estimation in an agile software development environment.

3. Effort Estimation.

One area in which the current research has fallen short has to do with linking the impact of technical debt with established theories related to effort estimation. As pointed out by Laqrichi, Gourc, and Marmier (Laqrichi et al. 2015), a study conducted by The Standish Group International found that 44% of software projects extended beyond the expected cost and time estimates. In the 1970s, well before agile software development was popularized, Boehm introduced a common strategy for effort estimation called COCOMO (Boehm 1981). His technique considers source lines of code (SLOC) as a primary factor for computing accurate estimates. There have been follow-on attempts and techniques offered, including COCOMO II (Clark et al. 1998). However, with the rise in popularity of agile software development techniques in the last twenty years, there has been a shift away from earlier effort estimation strategies in favor of techniques involving human perception, even asking the software developers to assist with the estimate of small components or iterations of an overall software project.

A complete example of an effort estimation approach in agile software development is provided by Coelho and Basu (Coelho and Basu 2012). The authors depiction of this effort is shown in Figure 2.6. In their paper, an agile software development project is defined in short iterations. For each iteration, the amount of work of the agile teams would be a combination of various features or tasks, where each task is known as a story. For each story, the effort must be estimated, which should incorporate the size, complexity and risk of the effort based on

knowledge of the task and historical data of similar efforts. Each story effort estimation is broken into a team specific unit of measure called a story point. Larger efforts to complete a task translates to larger story point estimates. Next, by adding all the story points together, the team has a sum of story points that are desired for this iteration. At this point, Coelho and Basu move to the overall effort estimation of the iteration, where agile team leaders or members use a combination of intuition, team member time allocation, and historical data to determine if all the desired features or stories can be completed during the iteration or not.

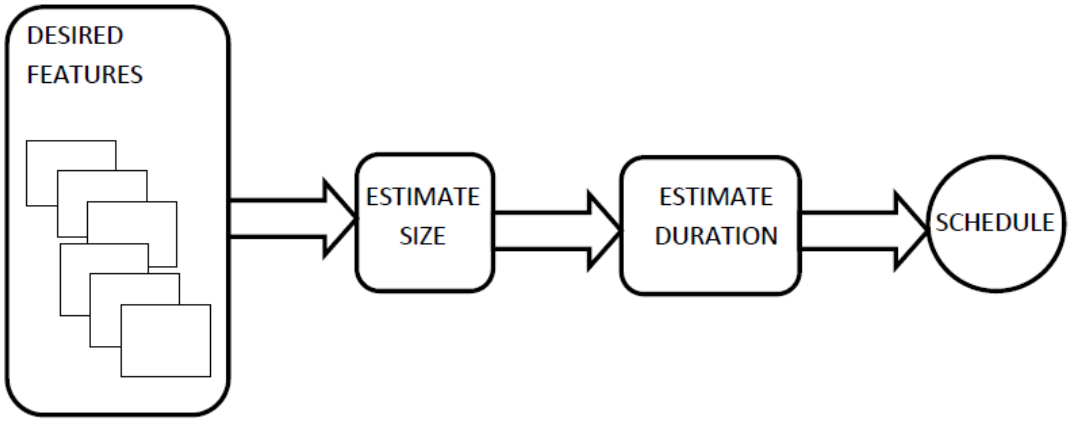


Figure 2.6. Overview of estimation effort for an agile iteration (Coelho and Basu 2012).

The above discussion presents just one approach to effort estimation in an agile software development project, albeit one of the more popular approaches according to Coelho and Basu. However, in the interest of avoiding somnolence, this dissertation will present research that attempts to summarize prior effort estimation studies involving agile software development initiatives.

Prior research in effort estimation in agile projects has been summarized by Usman et al. (2014). From a total of four hundred forty-three prior results found from their initial search of papers involving effort estimation in an agile software development environment, a total of

twenty-five previous studies were selected. Their research found a number of techniques used for effort estimation including expert judgement (e.g., the lead developer performs the effort estimate) and Planning Poker ("PlanningPoker.com - Estimates Made Easy. Sprints Made Simple." n.d.). The most popular techniques were those involving subject estimation. Further, a story point was one of the more frequently-used metrics for determining the size of an effort, reinforcing the statements made by Coelho and Basu. Measuring error was also discussed, where determining the difference between the actual effort time and the estimated effort time were necessary. Both the magnitude of relative error (MRE) and the mean magnitude of relative error (MMRE) were the most frequently used techniques for accuracy measurements. The authors also noticed that six of the studies did not include accuracy metrics. Further, team skills, prior experiences of team members, and task size were cited as the three important cost drivers for effort estimation in agile software development. In Usman's literature review, the conclusions provide additional impetus for this dissertation, specifically that, "Practitioners would have little guidance from the current effort estimation literature in ASD [Agile Software Development] wherein we have techniques with varying (and quite often low) level of accuracy (in some case no accuracy assessment reported at all) and with little consensus on appropriate cost drivers for different agile contexts. Based on the results of this SLR [systematic literature review] we recommend that there is strong need to conduct more effort estimation studies in ASD context that, besides size, also take into account other effort predictors."

Nguyen-Cong and Tran-Cao also conducted a review of effort estimation studies, but extended their review to a broader categorization of software development methodologies including agile, iterative and incremental software development (Nguyen-Cong and Tran-Cao 2013). A total of thirty-two research papers were chosen from publication databases including

ACM, IEE, Science Direct, and Springer. Their findings showed that model-based estimation (e.g., COCOMO, etc.), monitoring-based estimation (e.g., earned value management from Boehm and Turner 2005) and expert-based estimation (e.g., planning poker, etc.) techniques made up almost 60% of the estimation approaches. As far as determining the size metrics, story points followed by lines of code (LOC) were the two most popular strategies. As found in the previous literature review, MRE and MMRE were used most often for calculating accuracy. The authors also found that only half of the studies used empirical data to validate their estimation models.

Besides providing an overview of agile software development as well as a historical review of effort estimation techniques in existing agile and traditional software development environments, Ziauddin, Tipu and Zia provided additional granularity into the effort estimation strategies for agile software development as well as the costs of the software development project (Ziauddin and Zia 2012). Of particular interest to this dissertation is their "User Story Complexity Scale." This was an important attempt to give specific guidance to determine an appropriate story point value based on the specified guidelines. A noteworthy guideline involved the inclusion of refactoring complexity into the story points determination. This brief hint is one of the few research papers indicating the value of including the impact of technical debt as a factor into the effort estimation. Refactoring, or the "act of modifying software structure without changing its observable behavior," is covered in detail by Mäntylä and Lassenious (Mäntylä and Lassenius 2006). While the authors provide a taxonomy of drivers that indicate a need to refactor, it is important to note for this dissertation that refactoring indicators are not limited to the remediation of technical debt. Refactoring covers much broader categories that include variable naming convention changes, modularizing code, and security enhancements

to address new data loss risk to name a few. Returning to the research conducted by Ziauddin et al., refactoring was included in three of the five complexity scale categories, ranging from "significant" to "some" refactoring.

Finally, Laqrichi, Gourc, and Marmier discussed effort estimation models that integrate risk (Laqrichi et al. 2015). As mentioned previously, the authors reported that 44% of software projects exceed the cost and time estimates. In their findings, several factors were identified that may contribute to this effort estimation flaw. In their view, software risks are an unexplored factor of faulty effort estimation. In their words, "A software risk is an event that may or may not take place and that results in negative consequences on a software project." Some of the software risks included user resistance to change, unclear system requirements, immature technology, and organizational restructuring. In Figure 2.7, they provided a proposed model for risk identification along the path of the project lifetime. While their model does not address a chief concern of this dissertation, namely technical debt's impact on effort estimation, it does provide some guidance regarding the importance of making agile software development teams aware of key factors in order to reduce the impact of those factors on the effort estimation task, starting with the anticipation and identification of risk early in the agile software development effort.

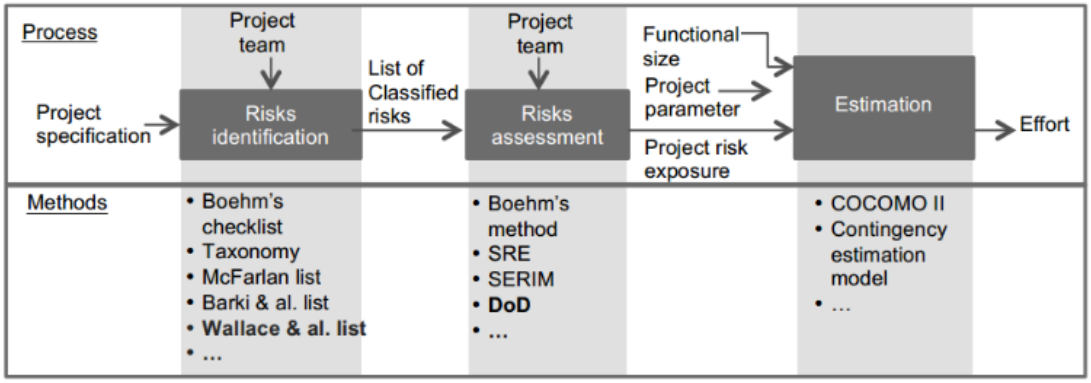


Figure 2.7. Integrating risks into an effort estimation process (Laqrichi et al. 2015).

2.6. Guidance for Further Study.

In brief review, this chapter has examined literature on five areas of research related to this dissertation: technical debt, agile software development projects, project or project management factors, team development factors, and effort estimation. Some key findings were identified by this review.

1. There is ample evidence that agile software development techniques are being studied, but authors consistently indicate that further study is warranted due to the lack of empirical studies.
2. Prior research into technical debt prevention and detection exists. These studies are almost exclusively tied to agile software development efforts as opposed to traditional software development techniques (e.g., waterfall).
3. Effort estimation continues to be of paramount importance to the software industry. As it relates to agile software development, prior research studies emphasize the subjective nature of effort estimation in an agile software development environment.
4. After a thorough review, a gap in the research has been identified as it relates to technical debt. Specifically, there appears to be no research model that incorporates technical debt into an existing effort estimation process involving agile software development projects.

As a result, this dissertation presents a model to develop further insights into accurately estimating agile software development efforts while incorporating technical debt knowledge as a key factor in the effort estimation process.

CHAPTER III
METHODOLOGY

The literature review in Chapter II provided the necessary prior research related to this dissertation topic. In that chapter, a number of studies involving effort estimation and the factors affecting effort estimation were identified, reviewed, and discussed. A gap in the prior research was identified involving the inclusion of technical debt in the effort estimation process for to agile software development models. While this exclusion may be considered minor by some researchers and practitioners, it is vital that no stone be left unturned while investigating this important topic.

The purpose of Chapter III is to articulate the research question of this dissertation, to propose a new model that answers the research question with formally stated propositions, and to provide a strategy for operationalizing the model. In the dissertation's model development effort, flowing from defining the dependent variable to determining the research methods to operationalizing the model, all leading to the empirical and theoretical conclusions, this dissertation will follow the guidance of Reithel's Conceptual Model Development Process (CMDP) shown in Figure 3.1 (Reithel 2009). In Figure 3.1, the blue arrows represent the recommended flow of model development. The colored arrows represent recognized opportunities to revisit and reconceptualize the research model as the research process exposes areas of refinement to improve the model or suggest areas of further research.

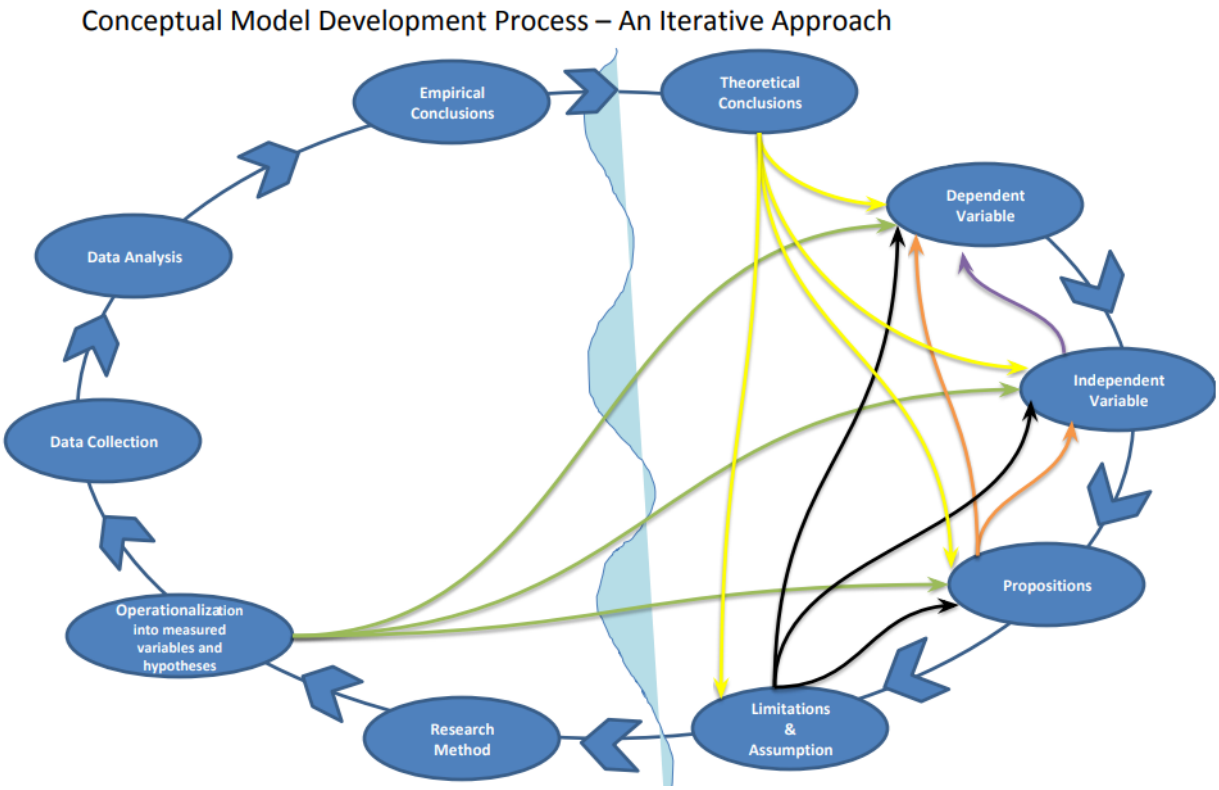


Figure 3.1. Reithel's visualization of the Conceptual Model Development Process (2009).

3.1. Research Question.

For multiple years, accurate effort estimation has been an elusive and complex topic to grasp in the software industry. The goal of this dissertation is to bring focus to the topic of accurate effort estimation specific to agile software development environments, one of the most commonly adopted software development models in use today. To that end, the dissertation's question is straightforward:

RQ: In an agile software project, does the consideration of technical debt during the effort estimation process improve the accuracy of the effort estimate?

In an attempt to add further clarity, the research question from the perspective of the practitioner is provided below.

RQ (Restated for Practitioner): If the agile team includes the possible impact of technical debt when providing time estimates of their effort, does the team avoid extending the estimated software development deadline for the project?

3.2. New Conceptual Model and Defining the Independent Variables.

See Figure 3.1 for a visual representation of the conceptual model developed in this dissertation. The model has been named the "Factors Influencing Agile Development Effort Estimation Accuracy" (FIADEEA). Before proceeding with a discussion of the conceptual model, it is important to note that this model includes the impact of a variety factors that impact an accurate effort estimation, not just technical debt. This gives greater validity and credibility to the model.

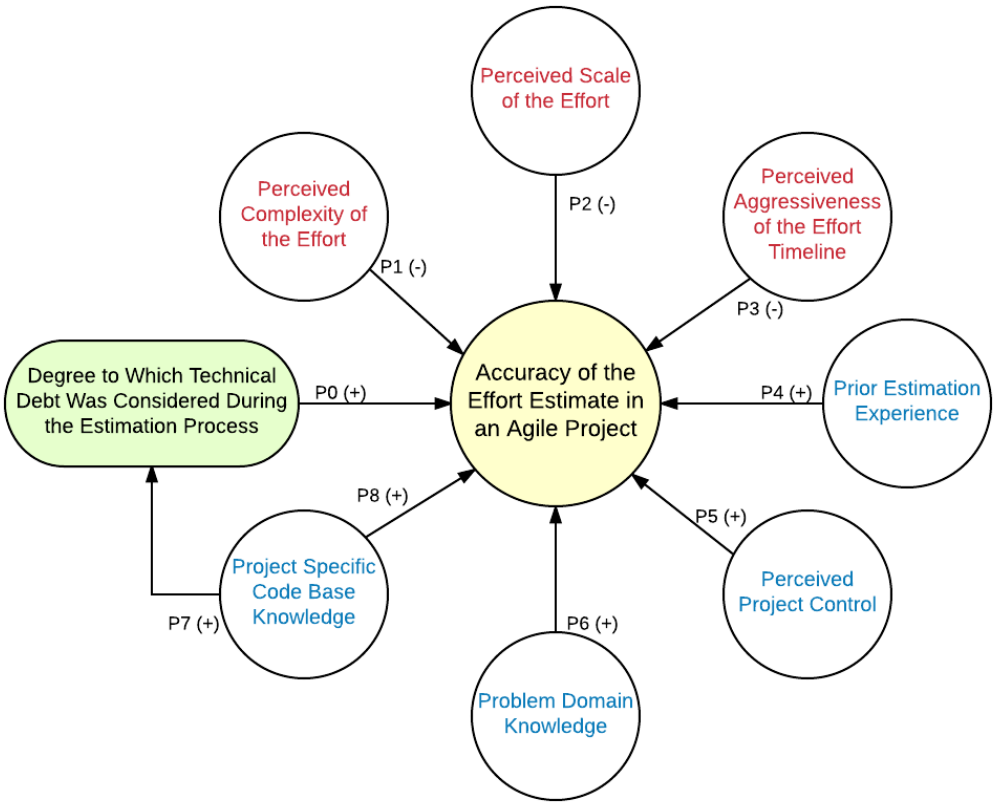


Figure 3.2. A visual representation of the FIADEEA model.

- At the center of the FIADEEA model is the dependent variable, the accuracy of the effort estimation in an agile project.
- To the left of the FIADEEA model is the key construct for this dissertation, the degree to which technical debt as considered during the effort estimation process.
- Along the top of the FIADEEA model, the constructus represent project-based characteristics important to study of this model. A project-based construct focuses on the software development effort overall. These factors are denoted by red lettering.
- Along the bottom of the FIADEEA model, the constructs represent team or team member-based characteristics important to the model. A team-based construct focuses on individual factors that influence effort estimation accuracy. These factors are denoted by blue lettering.

3.2.1. Conceptual Model Definitions.

Accuracy of the Effort Estimate in an Agile Project (AEE). AEE is located at the center of the model and is the dependent variable for this dissertation. AEE is the degree to which the estimated timeline of the effort concurs with the actual effort. By measuring the difference between the estimated effort time and the actual effort time, the effort estimation accuracy can be determined for an agile project. Prior research makes use of mean relative error (MRE) as well as mean magnitude of relative error when access to the observable data is available to the researcher. This dissertation intends to determine the accuracy of the effort estimate through feedback from survey participants and, as a result, will rely on the participants' feedback in determining their perspective of the accuracy of the effort estimate.

Degree to Which Technical Debt was Considered During the Estimation Process (TD). The degree to which agile team members specifically discussed and/or considered technical debt in the existing source code, libraries or modules during the effort estimation task. The agile team's familiarity with the concept of technical debt, the issues created by perpetuating and ignoring the influences of technical debt on a code base, and the impact of technical debt on effort estimation have the potential to impact the accuracy.

Prior research involving technical debt has focused on the identification, prevention and management of technical debt in an agile software development project. This provides value to the practitioners who recognize the negative impact of technical debt on the future development involving a software code base. And, while agile teams recognize the influence of technical debt on project delivery, prior to this research, technical debt has not been present in popular agile effort estimation techniques.

Perceived Aggressiveness of the Effort Timeline. The degree to which a project's timeline is compressed below the ideal timeline. For many software development firms, the timeline for project completion is determined by a number of influencers including the customer, business units and executive management, as well as the software development units. As it relates to faster revenue realization, time of delivery of a software effort is important to the business units of a software development firm as well as the client.

Perceived Complexity of the Effort. The extent to which the number of components and associated component interactions is considered challenging. During project or agile sprint planning, an agile team will develop a list of modifications necessary to meet the project deliverables. Effort difficulty and complexity in agile projects is routinely measured by story points or user points. An increase in the number of points corresponds to an increase in the

difficulty and complexity of the agile team's effort to deliver the particular component of the overall project. Prior research by Ziauddin et al. (Ziauddin and Zia 2012) emphasized the importance of complexity's impact on agile effort estimation. They suggested the use of a User Story Complexity Scale unique to the agile team, which would rank the effort based on a number of factors including research requirements, difficulty of judgement calls, system or subsystem dependencies as well as the need for refactoring.

Perceived Scale of the Effort. The size of the effort. Perceived scale is concerned with factors that influence the size of the effort including the projected budget costs, personnel count, geographic involvement, and project duration just to name a few. An example of a large-scale project would be the replacement of an accounting system at a Fortune 500 corporation.

During the literature review, scale was recognized as a factor that impacts agile projects. Most notably, Dikert et al. (Dikert et al. 2016) researched various large scale projects over a ten-year period. Their research found nine categories of challenges: integration of non-development functions, difficulties in implementing agile, resistance to change, requirements engineering challenges, coordination challenges in a multi-team environment, lack of time and financial investment, hierarchical management and organizational boundaries, quality assurance challenges, and different agile approaches in a multi-team environment. As for success factors, the combined list of team and individual success factors include management support, commitment to change, change leaders, customizing the agile approach, piloting, training and coaching, engage everyone, communication and transparency, mindset alignment, team autonomy, and requirements management.

Prior Estimation Experience. The number of software development projects that team members have participated in estimating. According to Usman et al, not all agile team members

use effort estimation.(Usman et al. 2014). While this dissertation is focused on agile projects, there is no intent to limit the evaluation of prior estimation experience to agile projects. As a result, the estimation experience may also include effort estimation experience in non-agile projects using strategies such as COCOMO ("COCOMO" 2017).

Perceived Project Control. The degree to which the effort timeline can be influenced. The belief by the agile team of having some influence on the project deliverables. For example, an agile team may desire to postpone new project enhancements brought forward by a client until a current agile effort known as a "sprint" is complete. Having project control means the software development team can influence the internal or external customer to postpone the enhancement request and avoid jeopardizing the delivery.

Problem Domain Knowledge. The degree to which agile team members are familiar with the discipline or field that the application encompasses. Throughout an agile project, agile team members will plan and develop solutions to address various programming problems. A combination of the agile team's skills, experience and familiarity (a.k.a., program domain knowledge) with each type of problem will guide the team towards specific solutions. Related, in 1983, Brooks published a theory on how programs are comprehended by programmers. A major point of his theory involves the process of "constructing mappings from a problem domain … into the programming domain" (Brooks 1983). Related, Ziauddin et al. proposed that agile teams are made up of "generalizing specialists" that has "at least a general knowledge of software development and the business domain in which they work" (Ziauddin and Zia 2012).

Project Specific Code Base Knowledge. The degree to which the agile team members have past experience with existing source code, libraries or modules that will be utilized in the software development effort. Effort estimation for an agile team may be more challenging for

unknown or unfamiliar code bases when compared to code bases developed by the agile team. Previously, Nasser performed a qualitative study to determine the factors that impact effort impact accuracy (Nasser n.d.). One of his named factors was the "knowledge and estimation of learning curve" which included the code base knowledge. In his words, "When developers are not familiar with technology or codebase, they are prone to underestimate or overestimate."

3.2.2. Conceptual Model Propositions.

While Figure 3.2 provides the granular detail necessary to show the various factors that influence the accuracy of the effort estimate, Figure 3.3 provides an alternative visualization which categorizes the factors by team and by project. The categorized model reveals that all team specific characteristics in the model should have a negative influence on the accurate of the effort estimation, while the project specific factors and the consideration of technical debt should have a positive influence on the accuracy of the effort estimate.

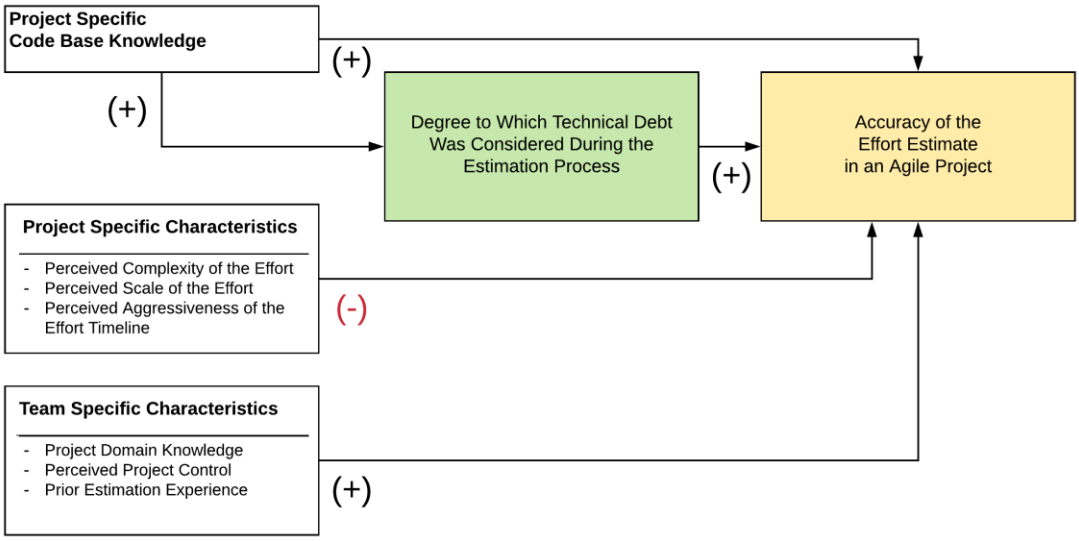


Figure 3.3. A visual representation of the model grouping team and project factors.

Having explained the conceptual model definition and provided an alternative visualization to help clarify the model, the next section covers the conceptual model propositions, which express the relationships between the constructs and the dependent variable, the Accuracy of the Effort Estimate in an Agile Project (AEE).

1. **P0:** *The Degree to Which Technical Debt was Considered During the Estimation Process (TD) is positively related to AEE.* This proposition is intended to be one of the key contributions of this dissertation. Prior research has not included TD in estimation models or efforts. This dissertation postulates that the consideration of technical debt (TD) will improve the effort estimate in an agile project.
2. **P1:** *The Perceived Complexity of the Effort is negatively related to AEE*. Consistent with prior research, this dissertation also suggests that effort complexity affects agile projects. As complexity increases, the inability for agile team members to "see" the impact of the component interactions will negatively influence the ability to accurately estimate the effort.
3. **P2:** *The Perceived Scale of the Effort is negatively related to AEE.* As mentioned in the definitions section, scale is the size of the effort. While some researchers may combine scale with complexity, this dissertation intentionally segregates the two constructs in order to attempt to isolate the influence of project size.
4. **P3:** *The Perceived Aggressiveness of the Effort Timeline is negatively related to AEE.* For individuals with a software development background as a practitioner, it "makes sense" that developers would argue that an increase in aggressive timelines would reduce AEE. This dissertation intends to include perceived aggressiveness as a

construct to determine its influence on AEE from the perspective of developers, technical project lead (e.g., scrum master) and other agile project team members.

5. **P4:** *The Prior Estimation Experience is positively related to AEE.* This dissertation expects seasoned software developers as well as seasoned software development teams have the edge on knowing how to perform software effort estimations and should give more accurate estimates.
6. **P5:** *The Perceived Project Control is positively related to AEE.* An increase in project control gives the agile team members increasing confidence that their project timeline estimates will not be negatively affected due to changing priorities or requirements.
7. **P6:** *The Problem Domain Knowledge is positively related to AEE*. When the agile team or team members have more familiarity with the business or technical problem which the software development effort intends to address, this dissertation assumes their ability to make accurate effort estimates increases.
8. **P7:** *The Project Specific Code Base Knowledge is positively related to TD.* When an agile team has developed significant portions of a particular code base, this dissertation expects their team's visibility into areas of potential technical debt to be high.
9. **P8:** *The Project Specific Code Base Knowledge is positively related to AEE*. This dissertation postulates that increases in code base knowledge increases the ability to accurately estimate the effort.

See Table 3.1 for a tabular view of the model propositions as well as their proposed relationships.

Proposition	Focus	Relationship	Description
Proposition 0 (P0)		+	The Degree to Which Technical Debt was Considered During the Estimation Process (TD) is positively related to AEE.
Proposition 1 (P1)	Project	-	The Perceived Complexity of the Effort is negatively related to AEE.
Proposition 2 (P2)	Project	-	The Perceived Scale of the Effort is negatively related to AEE.
Proposition 3 (P3)	Project	-	The Perceived Aggressiveness of the Effort Timeline is negatively related to AEE.
Proposition 4 (P4)	Team	+	The Estimation Experience is positively related to AEE.
Proposition 5 (P5)	Team	+	The Perceived Project Control is positively related to AEE.
Proposition 6 (P6)	Team	+	The Problem Domain Knowledge is positively related to AEE.
Proposition 7 (P7)	Team	+	The Project Specific Code Base Knowledge is positively related to TD.
Proposition 8 (P8)	Team	+	The Project Specific Code Base Knowledge is positively related to AEE.

Table 3.1. Proposition definitions and their relationship to the dependent variable.

3.2.3. Limitations and Assumptions.

This dissertation will diverge from the CMDP model outline shown in Figure 3.1 and postpone a discussion of the limitations and assumptions until the concluding chapter, in line with a typical dissertation format. In summary, this study targets software development practices within agile software development models. Alternative software development models, such as waterfall or iterative, are reserved for further study.

3.2.4. Research Method.

When evaluating the possible research methods, both qualitative methods (Myers 2008) and quantitative methods were considered. A factor that created pause in the research method selection process was the availability of local software firms that actively incorporate agile software development methods. Two software development firms were in two very different stages of corporate development. First, there is company A, a start-up company with approximately sixty employees focused on serving the needs of capital markets bank customers throughout the United States. Separately, company B with over three hundred employees has a twenty-year software development presence, having been purchased in a corporate acquisition in 2016. The software maturity at various stages of development at these two firms would improve the likelihood of the presence of technical debt and recognition of technical debt concerns among the agile practitioners.

With the availability of local software firms willing to cooperate with a study that could make progress in solving an important business problem facing all agile practitioners, a qualitative case study may be beneficial. However, the practicality of conducting numerous interviews and case studies would likely be met with management reservation with concerns of

impact to software deadlines and deliverables, a key concern of this dissertation. As a result, a quantitative research method was chosen utilizing an online survey hosted through Qualtrics.

3.3. Operationalizing the Model.

This dissertation presents a model to develop further insights into accurately estimating efforts of agile software development projects while incorporating technical debt knowledge as a key factor in the effort estimation process. The corresponding study will follow the suggested guidance of Churchill and his paradigm for developing measures (Churchill 1979), including specification of the domain construct and instrument development, collecting pilot study data and purifying the instrument, conducting a main study and assessing the reliability and validity, and making final empirical conclusions. Regarding the participants, this study will involve practitioners having experience with agile software development methods and models.

3.3.2. Domain Construct and Instrument Development.

Once a review of the relevant literature was complete, a survey instrument was designed to measure the key constructs defined in the model shown above in Figure 3.2. Terminology specific to software development was incorporated throughout the survey. However, specific programming language examples and paradigms were avoided to allow for participation of agile practitioners from non-programming areas of agile software development support, including project management and quality control.

Related descriptive and demographic items were also included in the instrument, including the target customer type of the agile software development effort. Since this study is concerned with consideration of technical debt during the effort estimation process, this

instrument item intends to expose any positive or negative influence the customer type may have on the effort estimation. The survey instrument can be found in APPENDIX A.

3.3.3. Purifying the Instrument.

To purify the instrument, steps were taken to evaluate the convergent validity and discriminant validity of the instrument items used during the pilot study. For this dissertation, Exploratory Factor Analysis (EFA) was utilized in the pilot study. Due to the uniqueness of this study and the lack of prior study data sets, EFA was chosen because it does not assume any *a priori* relationship with the constructs and, as a result, should aide in the validation of the survey instrument (Petscher et al. 2013).

Continuing to purify the instrument, reliability measures were taken utilizing Cronbach's alpha (Jarvis et al. 2003) measurements. Consistent with an acceptable level of measured reliability in academic research, the targeted alpha coefficients were 0.7 or greater.

3.3.4. Conducting the Pilot Study.

A pilot study was conducted at two independent software vendors (ISVs) where agile software methods are utilized on a daily basis during their software development life cycle (SDLC) efforts and where the potential or at least the recognition of the possibility of technical debt exists. The respondents were not be restricted to software developers (programmers who create, update and enhance software applications). Instead, participation was encouraged across the entire spectrum of agile software methodology stakeholders, including project management, quality control, and information technology to name a few. In short, the participants were part-time or full-time practitioners with prior agile experience.

It is worth noting the history of software development at the two ISVs and to identify potential areas where technical debt may exist. Company B was a business venture started by four faculty from a local university in the late 1990s. The founders originally hired two PhD graduates to begin development on the original product, a desktop-based platform written using Borland Delphi, a programming platform which had its roots in the Pascal programming language. Prior to the year 2000, company B expanded software emphasis into data and analytics grew into different development teams. To further support the data operation in 2000, company B acquired a company funded through a California cooperative. The acquired company's primary programming language was Cobol with an Oracle database. In the same time frame, Microsoft began the development of their programming language of the future, C#, as well as the development of their .NET framework, their next generation software platform for Microsoft Windows (Williams 2002). Like many ISVs with products hosted on Windows-based operating systems, company B chose to migrate to the latest Microsoft development technologies and engaged in a codebase migration to the .NET framework model. By 2010, company B had deployed over fifty custom systems and switched to an agile software development model. Separately, company A was formed as an internal incubator project and was later spun out as a separate ISV. Company A promotes a single, multi-tenant web application built on top of the latest Microsoft .NET platform. As an incubator-based organization, company A development utilized agile software development model from the onset. But as a start-up, company A's development organization has experienced three significant personnel changes which may contribute to the existence of technical debt.

Initially, agile leadership within the two ISVs was engaged with an IRB-approved communication strategy to determine the level of participation as well as a strategy for

communicating the study without creating alarm due to concerns over phishing emails. Once an approved communication technique was finalized, the potential respondents received a link to the online survey instrument. To respect the time of the respondents and improve the possibility of survey completion, the anonymous, online survey included thirty succinct questions aimed at measuring factors that influence the accuracy of the effort estimate. Besides basic demographic questions that provided an opportunity for some detailed descriptive statistical analysis, the instrument questions were intended to correlate with the key factors in the model shown in Figure 3.2.

Details of the pilot study results can be found in Chapter IV.

3.3.5. Conducting the Main Study.

After the refinement of the instrument at the conclusion of the pilot study, the main study was conducted. Regarding the respondents, agile practitioner meetups in the United States were engaged. A meetup provides the unique opportunity to engage agile developers from a variety of ISVs. Once the appropriate meetup participation approval was granted, the revised, online instrument was delivered through the meetup organization's communication tool of choice. Similar to the communication sensitivity necessary during the pilot study, because of the prevalent concerns of phishing techniques, the meetup coordinator was initially engaged to ensure communication with meetup members was approved and conducted in a manner consistent with the meetup's typical communication format and forum. The meetup groups chosen specifically for their agile focus to minimize the participation of participants with experience with software methodologies other than agile. As with the pilot study, the participants were part-time or full-time practitioners with prior agile experience, which include roles involving project management, quality control, and development to name a few.

For the main study, Exploratory Factor Analysis (EFA) and Cronbach's alpha measurements were taken from the data to evaluate validity and reliability of the data collected from the main study.

3.3.6. Empirical Conclusions.

Beyond basic analysis using descriptive statistics, multiple regression was used to perform data analysis on the main study (McCullagh 1980). Chapter IV will discuss the empirical conclusions in more detail.

CHAPTER IV
RESULTS

This chapter discusses operationalizing the conceptual model described in Chapter III. First, an examination of the pilot study results is provided, including some of the key details of the study participants and instrument refinement efforts. This, in turn, resulted in enhancements to the main study intended to improve overall data quality while adding clarity to the research conclusion.

In the final section of this chapter, the main study results are presented and discussed. The main study involved the collection of survey data from a revised instrument based on feedback from the pilot study effort. Besides measuring the reliability and validity, multiple regression was used to test the proposed hypotheses.

4.1. Pilot Study.

As discussed in previous chapters and detailed in Chapter III, the pilot study was conducted at two independent software vendors (ISVs) where an agile software methodology had been adopted. Separately, while a qualitative study was briefly considered and would have proved beneficial in uncovering details about perceptions surrounding AEE, a quantitative study was chosen to ensure the time impact of conducting a study in these corporate settings was not detrimental to the project deliverables or deadlines of the independent software vendors (ISVs) participating in the pilot study.

The survey instrument incorporated the eight primary constructs that impact the Accuracy of the Effort Estimate in an Agile Project (AEE), namely *Problem Domain Knowledge, Perceived Project Control, Prior Estimation Experience, Perceived Complexity of the Effort, Perceived Scale of the Effort, Perceived Aggressiveness of the Effort Timeline, Project Specific Code Base Knowledge, and the Degree to Which Technical Debt was Considered during the Effort Estimation Process (TD)*. As depicted in Figure 3.3, the survey instrument intended to ask participants about the team's perceptions of complexity, scale and timeline aggressiveness, not their individual perceptions. As a result, the questions were carefully phrased to extract team views instead of individual views.

Related to the study participants, a unique challenge of the pilot study was the inclusion of steps required to ensure a number of unique agile projects were part of the pilot study. With only 2 ISVs participating, the expected tendency or concern was that all participants would provide survey instrument feedback on the most recent agile iteration or sprint, which would not provide sufficient data to evaluate the constructs of the proposed model. With this anticipated concern, the decision was made to ask the primary contact at each ISV to assign a unique identifier to specific iterations or sprints which occurred in recent months. This unique identifier would be provided by the primary ISV contact with knowledge of the sprint or iteration participants. Then, during the survey, the participant would enter a specific code that had no meaning to the researcher but would internally aide in ensuring a unique iteration or sprint was addressed by each participant. Further, in the event that duplicate identifiers were uncovered during the survey data analysis phase, the data could be averaged or simply eliminated. Finally, the respondents were not restricted to software developers (programmers who create, update and enhance software applications). Instead, participation was encouraged across the entire spectrum

of agile software methodology stakeholders include project management, quality control, and information technology.

The initial pilot survey instrument was deployed online, utilizing Qualtrics ("The Leading Research & Experience Software" n.d.). There were 32 questions. The link to the survey was made available to the 2 ISVs, specifically to the agile software development leads who distributed the link internally within their respective organizations. By the conclusion of the survey, there were a total of 26 responses. Of the total survey participants, 24% were female, 64% were male and 12% preferred not to answer.

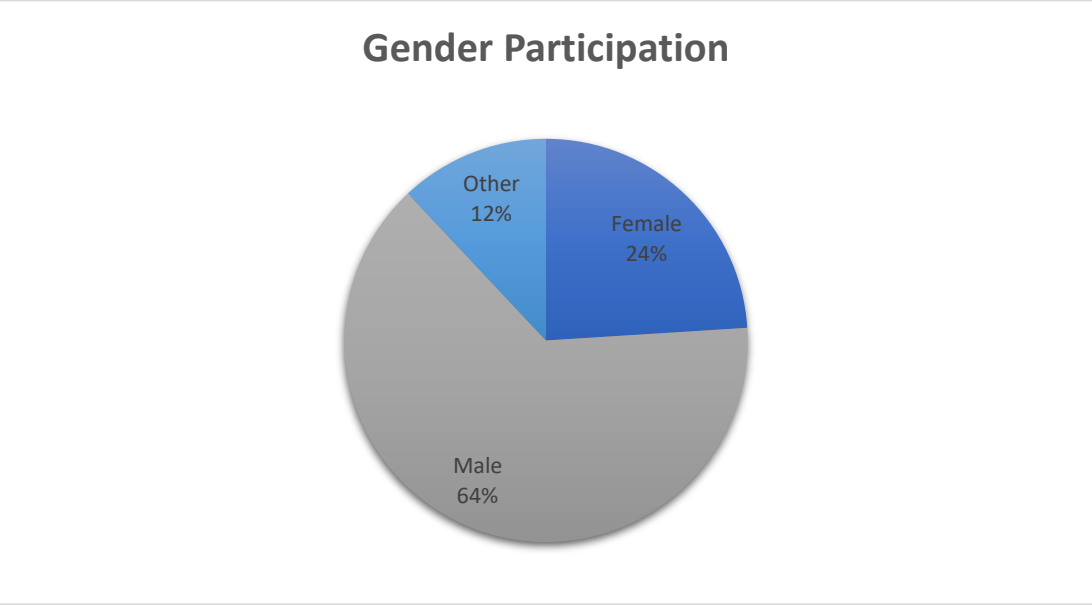


Figure 4.1. Gender Participation from the initial pilot.

When asked about the title that best describes their role, 40% answered with developer, 32% answered with tester, 8% answered as executive, 8% answered with product manager, 4% answered with personnel manager and finally 8% refrained from answering.

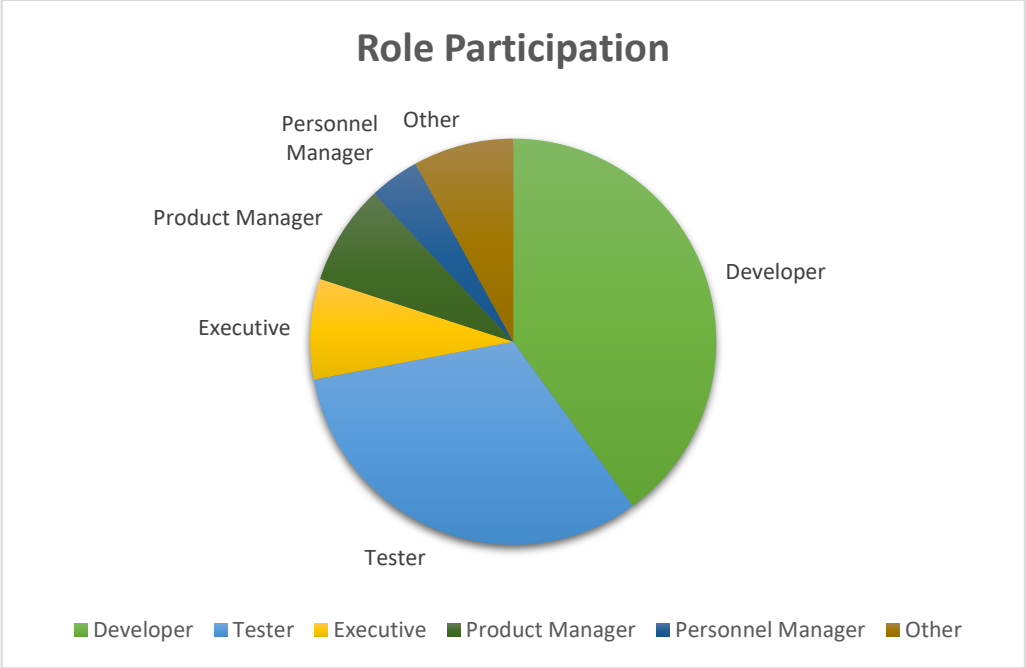


Figure 4.2. Project Role from the initial pilot.

Reviewing the question that targeted the dependent variable, "*For the particular iteration under consideration, how accurate was the effort estimation?*" the respondents were given a range from 1 (Inaccurate) to 10 (Accurate). The respondent's average response was 6.61 with a standard deviation of 2.00. While this does not indicate the cause of inaccurate effort estimation, it does indicate that agile team members recognize that the accuracy of the effort estimation is not ideal.

From the initial pilot survey, Cronbach's alpha measures for determining reliability were below 0.7 for some of the constructs. A primary cause was attributed to individual-focused questions instead of team-focused questions. For example, phrasing that asked "Are you" would indicate the request for the individual opinion. However, as mentioned in Chapter III, the goal was to ask the individual "Was the team" to gauge the team's perspective. As a result, the pilot study survey instrument questions were revised and the pilot study continued with a revised pilot

study. The questions in the revised pilot study can be found in Appendix A. Guidance was also provided to the ISV contacts to encourage participants to complete all questions.

The revised pilot survey instrument was also deployed online, utilizing Qualtrics ("The Leading Research & Experience Software" n.d.). There were thirty questions. The link to the survey was also made available to the 2 ISVs, specifically to the agile software development leads who distributed the link internally within their respective organizations. By the conclusion of the survey, there were a total of 29 total responses with 6 incomplete or duplicate responses bringing the total to 23 usable responses. Of the 23 participants, 13% were female, 87% were male.

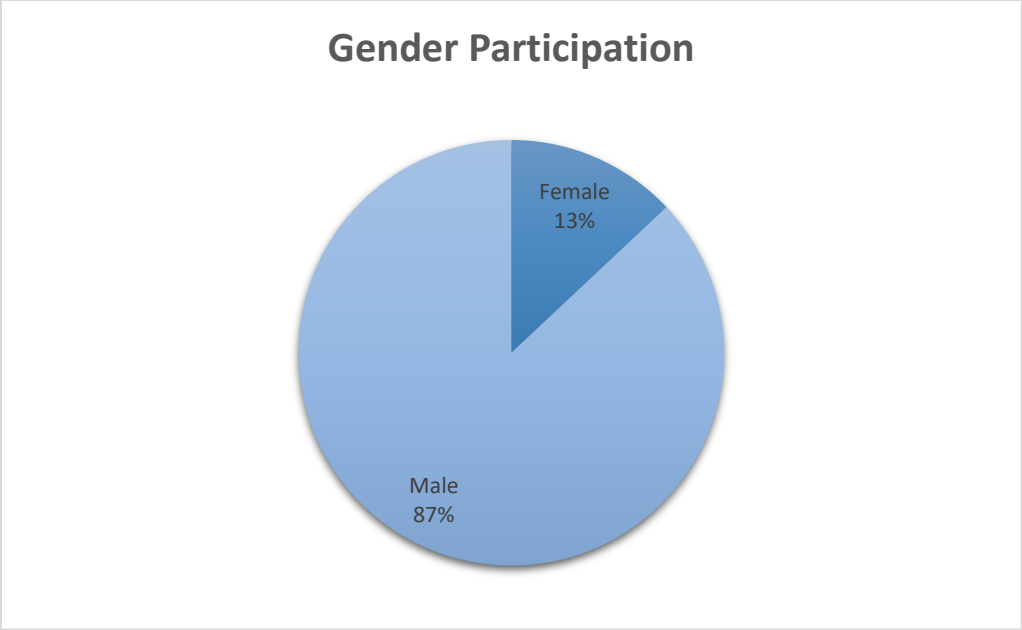


Figure 4.3. Gender Participation for the revised pilot.

When asked about their title that best describes their role, 56.5% answered with developer, 39.1% answered with tester, and 4.4% answered with product manager.

Figure 4.4. Role Participation for the revised pilot.

Reviewing the question that targeted the dependent variable, "*For the particular iteration under consideration, how accurate was the effort estimation?*" --- the respondents were given a range from 1 (Inaccurate) to 10 (Accurate). The respondent's average response on the revised pilot study was 6.74, with a standard deviation of 1.74. Once again, while this finding in the revised pilot survey does not indicate the cause of inaccurate effort estimation, it does indicate that agile team members recognize that the accuracy of the effort estimation is less than ideal.

The age of participants was also examined in both pilot studies. See Figure 4.5 for the age distribution for participants in the revised pilot study. While the career experience in an agile software environment of each participant was unknown, because the majority of the participants exceeded the age of 24 in these software development firms, effort estimation experience among participants is expected to be mature as compared to firms made up of primarily recent graduates.

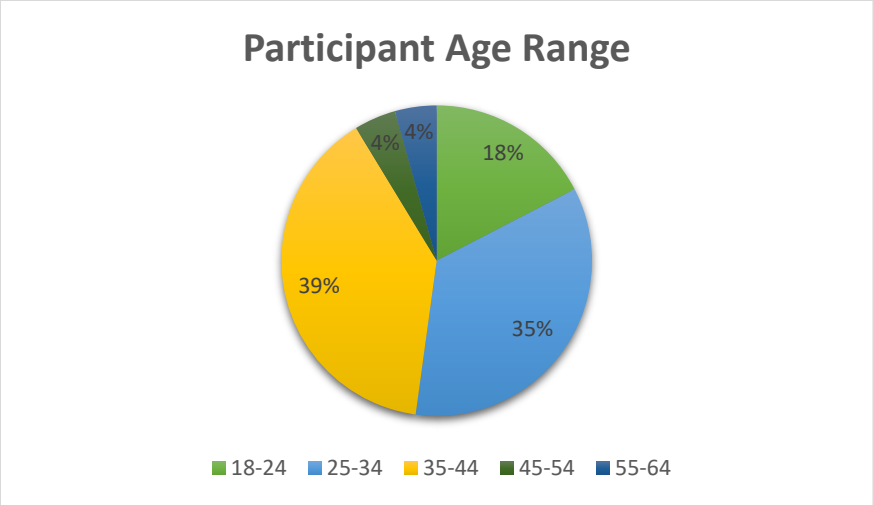


Figure 4.5. Age distribution of revised pilot participants.

As shown in Figure 4.6, this expectation is consistent with responses to one of the key construct questions, "*Rate your team's effort estimation experience."*

Figure 4.6. Participants rating of team's effort estimation experience.

For the project specific constructs, Figure 4.7 shows the responses for the key questions meant to determine perceived complexity, scale and timeline aggressiveness.

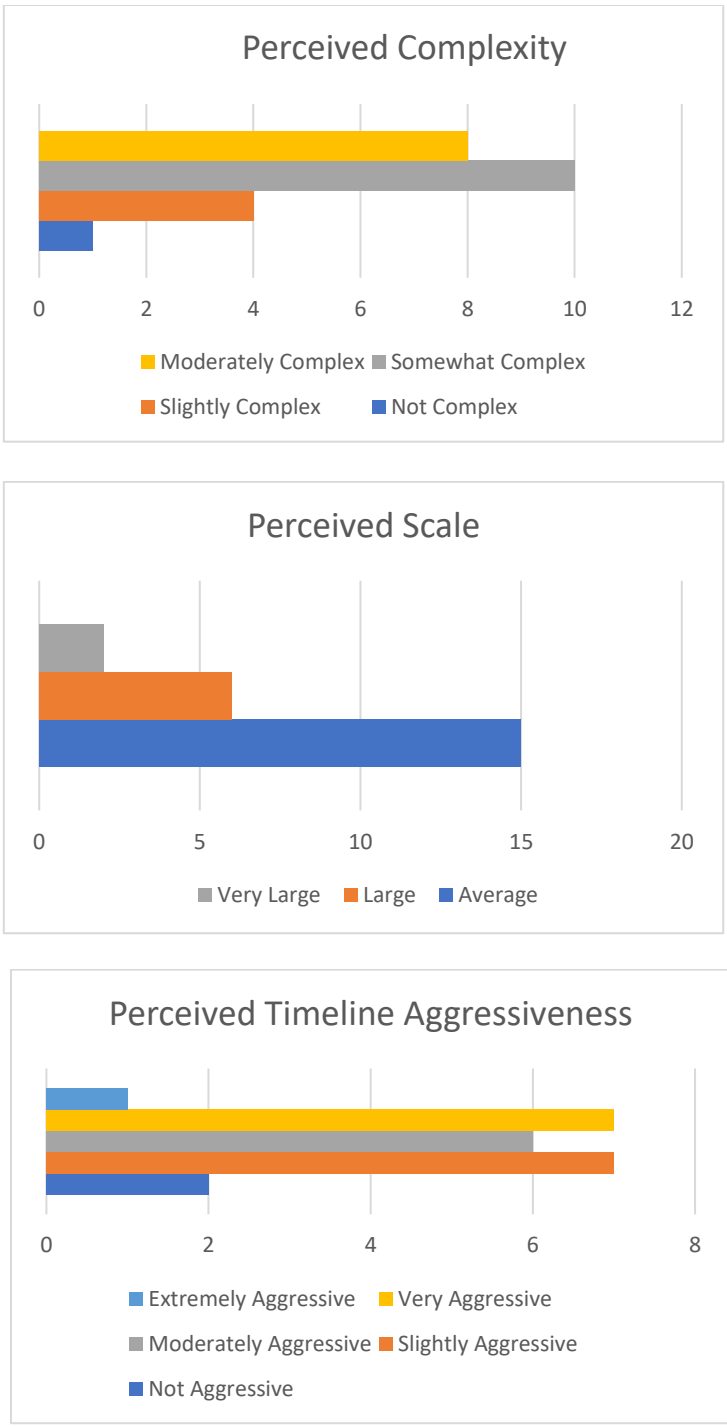
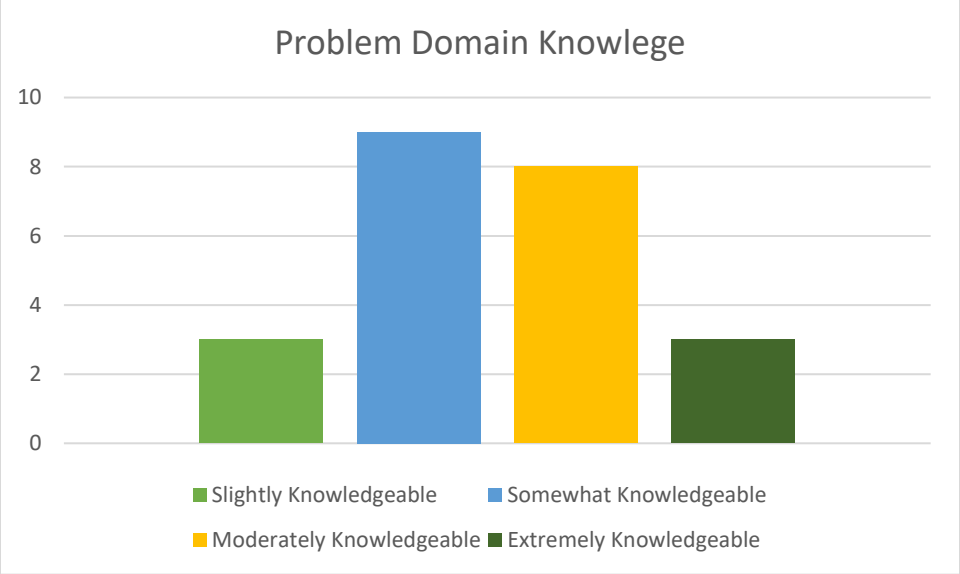


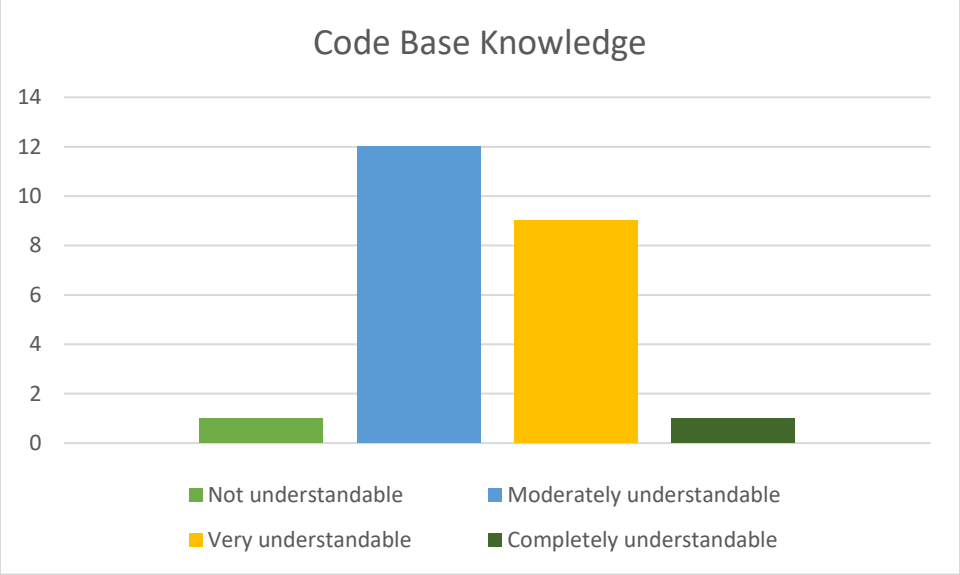
Figure 4.7. Responses to the project-based constructs.

The respondents revealed a consistent pattern of above average responses for all three constructs, with timeline aggressiveness being the only construct with an "extreme" data point,

depicting the aggressiveness of the timeline for the particular agile project under consideration for that respondent.

For the remaining team-specific constructs, Figure 4.8 shows the responses for the key questions meant to determine the problem domain knowledge, perceived project control and project specific code base knowledge for the teams.





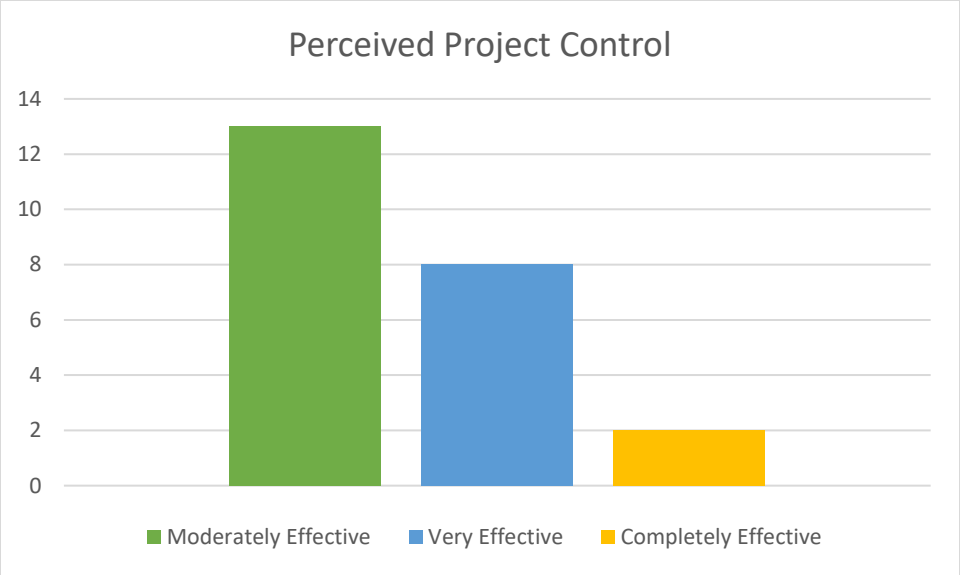


Figure 4.8. Responses to team-based constructs.

Once again, the respondents provided above average responses for all factors, suggesting the iterations were consistently made up of teams of knowledgeable developers, testers and managers.

When asked "*Was technical debt considered during the effort estimation process?*" --- the majority of respondents indicating varying levels of the degree to which technical debt was taken into consideration. See Figure 4.9 for a breakdown of the responses. While no conclusions can be made about the impact of technical debt consideration on the accuracy of an effort estimation during the revised pilot study, the varying responses do provide evidence that respondents were aware of technical debt and its potential influence on the effort estimation process.

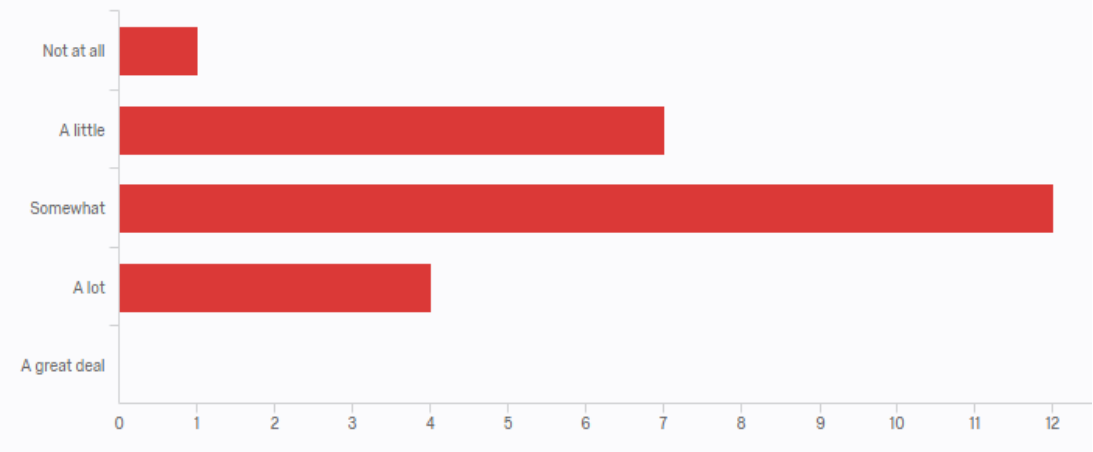


Figure 4.9. Consideration of Technical Debt during the effort estimation process.

Finally, when asked for open-ended feedback concerning actions to improve the accuracy of the effort estimate, there was significant feedback. See Table 4.1 for representative comments. Many of the suggestions could be categorized into the constructs identified in the FIADEEA model, including problem domain knowledge, complexity, perceived aggressiveness and prior estimation experience. Only a few comments indicated that the accuracy of the effort estimate was appropriate, echoing consistent results with the question, "*For the particular iteration under consideration, how accurate was the effort estimation?*" where the average score was 6.74 out of 10 in the revised pilot, with 10 representing a completely accurate effort estimate.

Was there anything the team could have done to improve the accuracy of the effort estimate?
The team in general has a lack of knowledge for the applications we support; increasing this would benefit the team greatly.
To complete the iteration, the tasks were very sophisticated and relied on a lot on other process knowledge. These layers of complexity need to be considered when denoting effort to tasks.
Could have identified requirements earlier that were identified during testing.
Effort should also account for scenarios that needs to be tested related to that particular code change as well as other area that might be affected, regression scenarios and previous experience about similar work item.
Possibly reviewed the work items before the start of the sprint so that we could have gotten a bit more familiar with current code structure.
The technical team need to work together to keep each other informed. The business team needs to talk to the technical team earlier in the project planning. Before committing dates and time lines the technical team needs to be talked to. By the time it hits the team business has already promised and realistic expectations were not given to the target user.
Our sprint estimation was pretty close to actual capability between dev and QA.
Be less ambitious regarding how much a single developer and/or QA can accomplish in 2 weeks.
The testing estimate should have been separate from the development estimate. When they're grouped together, it can be thrown off in accuracy in development, testing, or both. Separate estimates would be likely to cause less estimate issues for development.
At the time of this iteration our team had several very new members, making estimation difficult (lack of experience, code base knowledge, etc.). Our team is slowly getting better at estimation as we work on more items. We are also getting better at not committing to items with unfinished acceptance criteria and breaking down items that have too much. Really, time and small modifications to our process to see what works has helped us the most and will continue to. Giving people time to get used to the code base and work on a few items would've greatly improved estimates. Also, we often allowed our PO's to influence (rather, force) our estimates; we should've stuck to our guns and estimated what we felt was appropriate.
Need to understand business logic more.
I think our sizing is usually accurate. We rarely get bitten by technical debt we over looked. Other things like various project managers trying to get too many items in the sprint past what the scrum master set the velocity to, and various production issues that need looking into during the sprint are bigger problems.
To the new or junior developers, it is kinda hard to put an effort on some work items, but with the help from senior or mid-level devs, it is helpful. Breaking down big work items into small surely helped to estimate time more accurately.
A lot of the issues that have been encountered regarding complexity and sizing have been out of our control. The only thing we could have done better was temper expectations regarding the quality of the work that would be produced based on the timeline provided.
Include technical debt discussions in poker sessions and consider that in effort estimations.
The team could have had design meetings on how to tackle the problem outside of sprint planning or daily stand up. The team should be shortening stand up to be a meeting to remove blockers from development, but it tends to become long when we start talking about designing new items and the requirements for an item get edited and considered too much during a daily stand up. If we designed a solution instead of making assumptions about the code base we could initially spend more time coming up with a plan of action but delivering a more quality product.
More in depth discussion of issues. However, estimates for this sprint are more accurate than the team's recent sprints due to gaining familiarity with code base.

Table 4.1 Open-ended responses from the pilot studies.

4.2. Pilot Study Statistical Summary.

Due to the effort to revise the pilot questions, the revised pilot study results yielded improved results for the reliability of the constructs measured with Cronbach's alphas. Table 4.1 shows the initial findings.

Construct	Alpha on Standardized Items
Consideration of Technical Debt	0.691
Project Specific Code Base Knowledge	0.727
Problem Domain Knowledge	0.726
Perceived Project Control	0.143
Prior Estimation Experience	0.298
Perceived Complexity	0.306
Perceived Scale	0.713
Perceived Timeline Aggressiveness	0.693

Table 4.2. Measured reliability results of the revised pilot study responses.

Five of the eight constructs had alpha scores on standardized items above or near the desired minimum alpha cutoff of 0.7. Upon review of the three constructs that did not meet reliability expectations, further analysis noted phrasing in the corresponding survey questions that could be misinterpreted. As a result, the questions were revised for the main study.

Unfortunately, the revised pilot study sample size was insufficient to perform an exploratory factor analysis (EFA) across all eight factors using the revised pilot study data. But from the initial pilot responses, there were concerns that two factors, namely project complexity and project scale, could be considered a single factor by the survey participants. For the revised pilot study, modifications were made to the instrument to address the lack of clarity. Further, with the revised pilot data, a limited EFA was used to analyze the responses to survey questions targeting project complexity and project scale. This helped determine if the survey results indicated the existences of two separate factors or if a single factor was actually more suited for

this study. The EFA was performed on the six instrument questions involving project complexity and project scale. Table 4.3 shows the component matrix from the limited EFA analysis that was generated by SPSS. These results indicate the existence of two factors, with the separation of factors between project complexity and project scale, as originally intended in all but one of the instrument questions. The results of this EFA confirmed the need for separate factors for scale and complexity while providing insight into instrument items in need of revision. Specifically, for the final study, the instrument question, "*Rate the following components of your iteration*," will be revised to separate the complexity and scale questions from being physically adjacent to each other. While it may be expected that scale and complexity are correlated, the random question placement change was intended to help ensure that the survey respondents will evaluate complexity and scale as independent factors.

Rotated Factor Matrix[a]

	Factor 1	Factor 2
How would the team rate the scale of this project?	.995	.093
What was the team's perception of the scale of the project?	.718	.087
Rate the following components of your iteration. - Project Scale	.286	.834
Rate the team's perception of the complexity of this effort.	-.070	.560
Indicate your level of agreement with the following statement: The complexity of this iteration was excessive	.227	.301
Rate the following components of your iteration. - Effort Complexity	.043	.316

Extraction Method: Maximum Likelihood. Rotation Method: Varimax with Kaiser Normalization.

a. Rotation converged in 3 iterations.

Table 4.3 Factor Analysis of project scale and project complexity instrument questions.

Finally, Pearson correlations (Weaver and Wuensch 2013) were performed using SPSS, comparing the eight constructs against the output variable AEE. In Table 4.4, the correlation "row" between the constructs and the dependent variable AEE is shown. For purposes of the correlation computation, each construct represents a sum of the survey items related to the specific construct. In the Pearson Correlation row, items marked with an asterisk (*) denote significance at the 0.05 level (2-tailed).

		AEE	TD	PSCBK	PDK	PPC	PEE	PC	PS	PTA
AEE	Pearson Correlation	1	-0.096	.359	.468*	-0.102	.450*	-.268	-0.058	-.484*
	Sig. (2-tailed)		0.663	0.092	0.024	0.642	0.031	0.216	0.792	0.019
	N	23	23	23	23	23	23	23	23	23

Table 4.4. Correlation Matrix of the output variable row.

While it is premature to reach conclusions about the relationships and directional relationship between each independent variable and the output variable AEE, there are early indications that a number of the constructs exhibit construct validity and divergent validity consistent with the model. For the constructs that did not exhibit significance, this information, in combination with the reliability measures, was used to enhance the survey instrument prior to the main study.

4.3. Main Study.

The remaining steps of this dissertation's main study involve the explanation of the final operationalization of the FIADEEA model, involving the distribution of the final survey instrument found in APPENDIX C, and the corresponding data collection and statistical analysis, as well as the presentation of the results and research conclusions. Similar to the pilot studies previously performed, the targeted subjects were practitioners, specifically software professionals

experienced working in an agile software development environment. Survey subjects were targeted across the United States among agile software development colleagues in various corporate settings and agile "meetup" groups in major software hubs in Alabama, Texas, and Utah.

4.3.1. Operationalization.

The operationalization of the main study began as an extension of the two pilot studies. Specifically, a Qualtrics online survey was utilized to gather expert opinions. Because software professionals are comfortable with online or Internet activity, paper surveys were not introduced in the main study. Revisions to the online study items were made at the end of the pilot studies, primarily based on preliminary results of the measured reliability using Cronbach's alphas and feedback from the pilot participants. Due to the expectation that agile software professionals have limited time for survey participation, the final survey was limited to 32 questions. That limited the study to three questions for each FIADEEA model construct. See APPENDIX C for the final instrument.

4.3.2. Gathering Data.

In order to gather sufficient data to test the FIADEEA model across multiple statistical techniques, the objective of the survey distribution was to gather 150 completed survey responses from agile software professionals across the United States. However, United States citizenship information was not required and consequently not collected in this study.

Consideration was given to the data collection techniques to ensure the quality necessary for this study. Since the objective of this dissertation is to introduce a model that would explain behavior found among agile software professionals, corporate agile professionals were the target respondent audience. Respondents were initially contacted by email, by corporate engagement

and by other reputable online social media resources where agile professionals are known to assemble online. This include both LinkedIn and Meetup.

Careful consideration was given to the use of online panels, particularly regarding the survey response quality. As pointed out by Roulin, these "so-called convenience samples (e.g., crowdsourcing, online panels, and student samples)" are not to be dismissed as potential data resources if the researcher will carefully examine the advantages and risks to accept or reject the data originating from these data sources (Roulin 2015). The online panel sources for agile professionals used in this study included Qualtrics Research ("The Leading Research & Experience Software" n.d.) and Positly ("Positly: Study Recruiter" n.d.). Based on the higher risks outlined by Roulin and the highly targeted group of software professionals, generic crowdsourcing techniques and tools such as Amazon Turk were not considered to be appropriate for this study. However, both Qualtrics and Positly employed techniques for pre-screening respondents to address quality and completeness concerns as well as ensuring the individual was an agile software professional. Qualtrics Research, being a leader in academic and professional study solutions, already had an established panel of software professionals. The additional pre-screening was a Qualtrics-internal process to ensure the respondent had experience in an agile software professional setting. Positly also has existing panels of software professionals and offered a more granular set of questions to further ensure the quality. First, Positly offered a break-down by software professions that include careers in information technology, budgeting, quality control, software development, networking. From that list, appropriate software professionals were selected for pre-screening. Second, Positly offered a creative screening strategy to identify agile professionals. See below.

Which of the following practices have you used in your professional work? Check all that apply.

[_] Agile Project Management

[_] Scrum Methodology

[_] 360 Degree Feedback

[_] Efficient Viral Actualization

[_] Cross-platform Unit Facilitation

[_] None of these

In the responses above, agile professionals would notice the first three choices associated with agile software development. However, the fourth and fifth responses were random phrases and not associated with any software development strategy. As a result, respondents who selected those two choices would be screened out of the study.

As a final screening strategy, there were two questions embedded in the main study intended to further identify agile professionals. First, the only narrative question proved useful in screening low quality responses. Question 31 reads, "*Was there anything the team could have done to improve the accuracy of the effort estimate? Please explain."* While question 31 was optional, a number of respondents in the pilot and main studies shared their feedback. When no feedback was offered, and no other screening triggering events occurred, the response was accepted as we would accept feedback from question 31 that was related to the question. However, responses to question 31 that were unrelated (e.g., "*it is easy to use and modern*") or simply random character typing (e.g., "*Y7v7v6v g6v v v7v7v7v*") were screened out of this study.

Second, question 12 required the respondent to identify a previous or current agile or sprint iteration. *"In order to group responses pertaining to the same iteration or sprint, please provide the first and last initial of the agile team lead followed by the end date ("MMDDYYYY") of the*

iteration or sprint (e.g., TC04162018).” Based on a brief analysis of the responses, question 12 proved to be a primary survey abandon point for partial responses and helped eliminate unqualified survey responses that were non-sensical. Finally, question 12 provided specific instructions in identifying a sprint by requesting a unique identifier but without compromising the anonymity of the participants. When multiple practitioners completed the survey for the same sprint, the unique identifier was be utilized to identify the duplicate responses and screen them from consideration for the purposes of this dissertation. However, the data was be retained for future studies.

For the main study, there were 293 partial and complete responses within the Qualtrics survey. After applying the screening techniques mentioned above, the final data set contained 150 valid responses (51.19% of all Qualtrics survey participants). Below are descriptive statistics about the respondents.

1. Using the longitude and latitude statistics provided through the Qualtrics survey, the approximate Internet location of the participants was determined. In all, 148 of the valid responses were from a United States based location. See Table 4.5.

United States	146
Canada	1
Netherlands	1

Table 4.5. Breakdown of respondents by country based on Internet location.

2. Regarding the respondents’ current corporate role, developers represented the highest role at 43.33%. For the top three participants by corporate role, next were project managers (17.33%) and product managers (12.67%). See Figure 4.10.

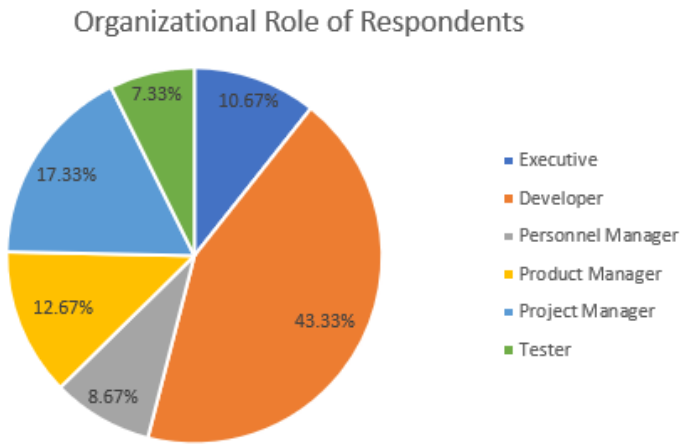


Figure 4.10. Percentage breakdown of respondents based on organizational role.

3. Regarding gender, 62.67% were male respondents and 37.33% were female respondents among the completed surveys used for this analysis. Further gender analysis against all Qualtrics respondents regardless of their completion status revealed a fallout rate of 16.38% for male respondents while the fallout rate for female respondents was only 13.65%. As expected, the inclusion of partial or incomplete responses accounted for approximately 17.5% with no gender response along with the remaining fractional response associated with "I prefer not to answer" or "Other." See Table 4.6.

   According to a 2014 survey conducted by Deloitte, Datawheel and Cesar Hidalgo, the gender composition of software developers, applications and systems software is 80.9% Male ("Software Developers, Applications & Systems Software" n.d.). While the deviation in this dissertation's respondents may be partially explained by software professionals who may not be primarily developers, including project managers, product managers, testers and executives, further analysis would be required to determine the potential source of the deviation. The finding merits further review and study.

Gender	Completed Surveys	Overall Fallout Rate
Male	62.67%	16.38%
Female	37.33%	13.65%

Table 4.6. Gender breakdown of respondents.

4. Age and software development experience was also captured. Respondents had an average age of 38 with an average of 12 years of software development experience. Figure 4.11 shows a histogram which breaks down the software experience data gathered during this study.

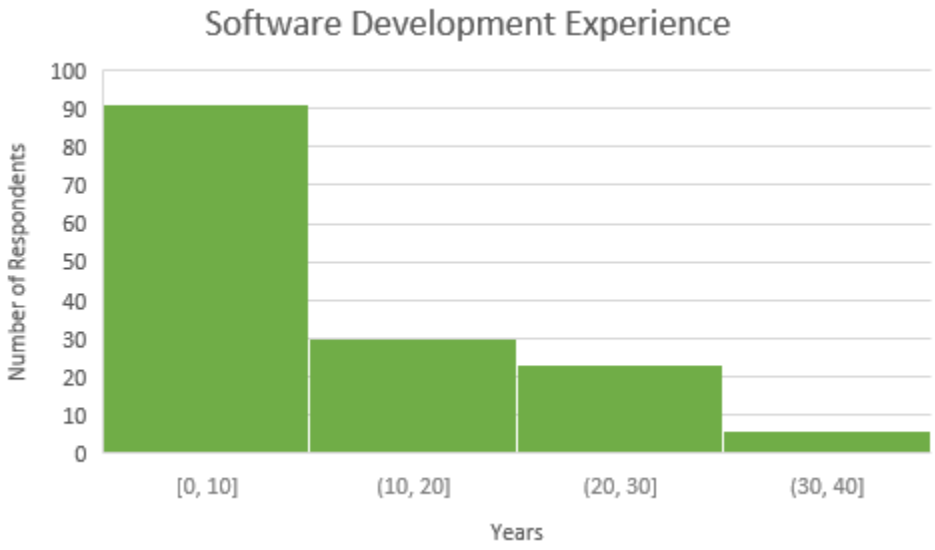


Figure 4.11. Histogram of software development experience among respondents.

5. A key component of this dissertation is the consideration of technical debt during the effort estimation process. When asked, "*Was technical debt considered during the effort estimation process?*" varying levels of technical debt consideration were reported during the main study. Figure 4.12 shows the frequency of responses across main study participants.

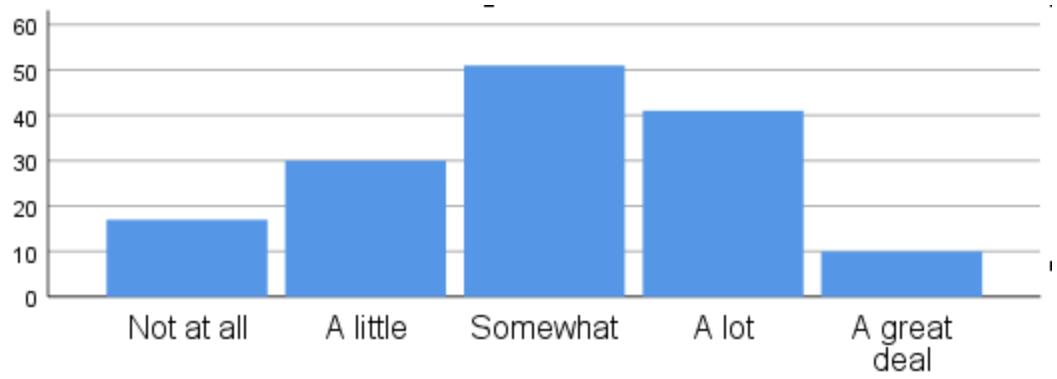


Figure 4.12. Consideration of Technical Debt during the effort estimation process.

4.3.3. Reliability.

Consistent with the effort during the pilot study, the reliability analysis in this study utilized Cronbach's alpha as the primary indicator of reliability of the data. Due to the effort to revise the pilot questions, the main results showed improved results for the reliability of the constructs measured with Cronbach's alphas. Table 4.7 shows the summary findings from the main study.

Construct	Alpha on Standardized Items
Consideration of Technical Debt	0.840
Project Specific Code Base Knowledge	0.747
Problem Domain Knowledge	0.732
Perceived Project Control	0.701
Prior Estimation Experience	0.730
Perceived Complexity	*0.648*
Perceived Scale	0.703
Perceived Timeline Aggressiveness	0.876

Table 4.7. Measured reliability results of the main study.

Seven of the eight constructs had Cronbach's alpha on standardized items exceeding the preferred alpha cutoff of 0.7. Perceived Estimation Experience required the elimination of one survey item to meet the alpha cutoff of 0.7. While Perceived Complexity did not meet the preferred alpha cutoff, the effort to improve the reliability after the second pilot resulted in a 100% improvement in the alpha measurement.

Overall, the reliability analysis demonstrated that the main study possesses adequate levels of reliability in all but one of the items. As noted by Tavakol and Dennick, a possible explanation for a low alpha is a low number of instrument questions targeting the construct (Tavakol and Dennick 2011). For Project Complexity, additional regression analysis will be included in later sections of this dissertation to expose the impact of Project Complexity in the FIADEEA model.

4.3.4. Validity.

Because sufficient data was collected during the main study, a factor analysis can be used to evaluate the data's convergent and discriminant validity. In the pilot studies, there was insufficient data to perform an Exploratory Factor Analysis (EFA). In the main study, sufficient respondent data was gathered and therefore a Maximum Likelihood Estimation technique within SPSS was used to perform an EFA. See Figure 4.13 for the factor loadings.

	1	2	3	4	5
PSCBK (Q22)	0.753				
PSCBK (Q29)	0.724				
PDK (Q14)	0.637				
PDK (Q25)	0.611				
PPC (Q18)	0.598				
PPC (Q21)	0.559				
PDK (Q24)	0.558				
PC (Q36)	0.455				
PEE (Q34)	0.440				
PTA (Q31)		0.897			
PTA (Q17)		0.809			
PTA (Q19)		0.740			
PC (Q30)					
TD (Q28)			0.873		
TD (Q33)			0.781		
TD (Q26)			0.669		
PS (Q15_4)				0.643	
PSCBK (Q15_2)				0.623	
PC (Q15_1)				0.601	
PPC (Q15_3)				0.508	
PS (Q23)					0.606
PS (Q16)					0.574

Figure 4.13. Exploratory Factor Analysis (Extraction: Maximum Likelihood, Rotation: Varimax with Kaiser Normalization).

The EFA revealed factor loadings for Factors 2, 3 and 5 that are consistent with the instrument's design, namely Timeline Aggressiveness (Factor 2), Technical Debt (Factor 3) and Project Scale (Factor 5) respectively. As for the noteworthy validity challenges, Factor 1 includes both Problem Domain Knowledge (PDK) and Project Specific Code Base Knowledge (PSCBK) instrument responses, meaning there were factor loading problems involving both PDK and PSCBK. It is important to recognize that both factors involve knowledge constructs, with PDK representing domain knowledge and PSCBK representing specific code base knowledge.

Separately, Factor 4 combined the survey items that were associated with 4 different constructs. Questions Q15_1, Q15_2, Q15_3, and Q15_4 covering 4 different constructs were visually grouped into a common presentation matrix within the survey instrument for the main study, as shown in Figure 4.14.

Finally, Q30, a factor corresponding to PC, did not meet the minimum factor location cutoff of 0.4. The combination of these loading problems represents the primary issues impacting the convergent validity of the FIADEEA model.

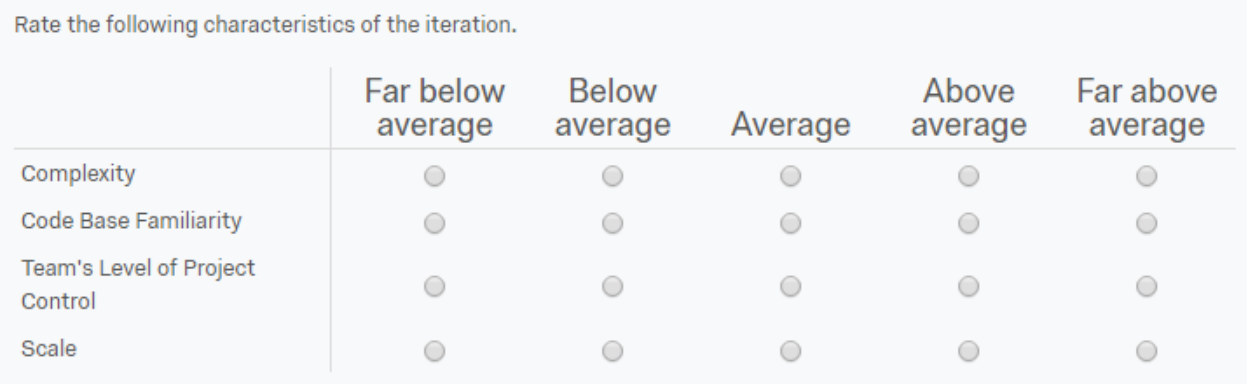


Figure 4.14. Question 15 from the main study showing the matrix grouping.

Completing the summary for the EFA, the corresponding eigenvalues are shown in Figure 4.15 from the first EFA. Consistent with the default for SPSS, eigenvalues above 1.0 was the cutoff for determining factors in the EFA analysis.

Factor	Total	% of Variance	Cumulative %
1	6.831	31.051	31.051
2	2.679	12.179	43.230
3	1.794	8.156	51.386
4	1.580	7.181	58.566
5	1.179	5.361	63.927
6	.990	4.500	68.427
7	.851	3.870	72.297
8	.739	3.359	75.656

Figure 4.15. Initial eigenvalues associated with the EFA.

Next, a factor analysis was performed with a fixed number of factors set to eight (8), which mirrors the number of independent variables in the FIADEEA model. Figure 4.16 shows these factor loadings for the main study revealing all loadings greater than 0.4. In this factor analysis, Factor 1 continues to be dominated by PDK and PSCBK. Also, Factor 2 and Factor 3 map to PTA and TD respectively.

	1	2	3	4	5	6	7	8
PSCBK (Q29)	0.779							
PSCBK (Q22)	0.777							
PDK (Q14)	0.617							
PDK (Q25)	0.586							
PDK (Q24)	0.577							
PPC (Q18)	0.473							
PTA (Q31)		0.891						
PTA (Q17)		0.796						
PTA (Q19)		0.749						
PC (Q30)								
TD (Q28)			0.893					
TD (Q33)			0.761					
TD (Q26)			0.669					
PS (Q15_4)				0.742				
PC (Q15_1)				0.559				
PSCBK (Q15_2)	0.458			0.475				
PS (Q23)					0.741			
PS (Q16)					0.583			
PEE (Q34)						0.919		
PEE (Q36)						0.446		
PPC (Q21)							0.93	
PPC (Q15_3)				0.416				0.734

Figure 4.16. Factor analysis, 8 fixed factors (Extraction: Maximum Likelihood, Rotation: Varimax with Kaiser Normalization).

The initial eigenvalues are shown in Figure 4.17 from the second EFA, where five (5) factors exceeded an eigenvalue of 1, which is consistent with the EFA findings shown in Figure 4.13.

Factor	Total	% of Variance	Cumulative %
1	6.831	31.051	31.051
2	2.679	12.179	43.230
3	1.794	8.156	51.386
4	1.580	7.181	58.566
5	1.179	5.361	63.927
6	.990	4.500	68.427
7	.851	3.870	72.297
8	.739	3.359	75.656

Figure 4.17. Initial eigenvalues associated with 8 fixed factors.

In Figure 4.16, utilizing a simple horizontal line test, visual evidence of discriminant validity for the majority of instrument questions is shown. That is, for each factor, a horizontal observance shows one and only one value. The noted exceptions are Q15_2 (Project Specific Code Base Knowledge) and Q15_3 (Perceive Project Control). Also, since Q30 (Perceived Complexity) did not meet the required factor cutoff limit of 0.4, no validity conclusions regarding it can be made.

Figure 4.18 shows the correlations of the 8 factors against Q13 which represents the Accuracy of the Effort Estimate (AEE). Of the 8 factors, six (6) were significantly correlated to AEE based. Further evidence of the correlation of PDK and PSCBK is demonstrated by the correlation between the two factors at 0.663, significant at the 0.01 level.

Correlations

		Q13	TD	PEE	PC	PSCBK	PDK	PTA	PS	PPC
Q13	Pearson Correlation	1	.422**	.460**	.031	.301**	.427**	.080	.227**	.438**
	Sig. (2-tailed)		.000	.000	.709	.000	.000	.330	.005	.000
	N	150	149	150	149	149	150	150	150	149
TD	Pearson Correlation	.422**	1	.380**	.270**	.268**	.280**	.305**	.327**	.292**
	Sig. (2-tailed)	.000		.000	.001	.001	.001	.000	.000	.000
	N	149	149	149	148	148	149	149	149	148
PEE	Pearson Correlation	.460**	.380**	1	.279**	.431**	.439**	.138	.395**	.444**
	Sig. (2-tailed)	.000	.000		.001	.000	.000	.092	.000	.000
	N	150	149	150	149	149	150	150	150	149
PC	Pearson Correlation	.031	.270**	.279**	1	.290**	.294**	.433**	.580**	.196*
	Sig. (2-tailed)	.709	.001	.001		.000	.000	.000	.000	.016
	N	149	148	149	149	149	149	149	149	149
PSCBK	Pearson Correlation	.301**	.268**	.431**	.290**	1	.663**	.180*	.406**	.554**
	Sig. (2-tailed)	.000	.001	.000	.000		.000	.028	.000	.000
	N	149	148	149	149	149	149	149	149	149
PDK	Pearson Correlation	.427**	.280**	.439**	.294**	.663**	1	.275**	.367**	.506**
	Sig. (2-tailed)	.000	.001	.000	.000	.000		.001	.000	.000
	N	150	149	150	149	149	150	150	150	149
PTA	Pearson Correlation	.080	.305**	.138	.433**	.180*	.275**	1	.466**	.121
	Sig. (2-tailed)	.330	.000	.092	.000	.028	.001		.000	.140
	N	150	149	150	149	149	150	150	150	149
PS	Pearson Correlation	.227**	.327**	.395**	.580**	.406**	.367**	.466**	1	.396**
	Sig. (2-tailed)	.005	.000	.000	.000	.000	.000	.000		.000
	N	150	149	150	149	149	150	150	150	149
PPC	Pearson Correlation	.438**	.292**	.444**	.196*	.554**	.506**	.121	.396**	1
	Sig. (2-tailed)	.000	.000	.000	.016	.000	.000	.140	.000	
	N	149	148	149	149	149	149	149	149	149

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

Figure 4.18. Correlations.

4.3.5. Regression.

As noted in section 4.3.3., the acceptable reliability of the model (as measured by Cronbach's alpha) as well as the research interest in evaluating the FIADEEA model performed meant that an evaluation of the regression model with eight (8) fixed factors would be beneficial in this initial study. However, due to the low Cronbach's alpha cutoff of the reliability measure on the Project Complexity (PC) construct (< .7), three regressions were performed with the combined PC predictor as well as with the two individual items predicting PC to account for the

possibility of impact to the overall model. See Figure 4.19 for the results of all three regression models. Note the "Predictors" footer, which indicates which item was used to predict PC. In all cases, the FIADEEA model was found to be significant with an R^2 no less than .377.

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics				
					R Square Change	F Change	df1	df2	Sig. F Change
1	.630[a]	.397	.363	1.722	.397	11.452	8	139	.000

a. Predictors: (Constant), PC, PPC, TD, PTA, PEE, PDK, PS, PSCBK

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics				
					R Square Change	F Change	df1	df2	Sig. F Change
1	.614[a]	.377	.341	1.752	.377	10.503	8	139	.000

a. Predictors: (Constant), Q30, PSCBK, TD, PTA, PEE, PPC, PS, PDK

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics				
					R Square Change	F Change	df1	df2	Sig. F Change
1	.640[a]	.410	.376	1.705	.410	12.061	8	139	.000

a. Predictors: (Constant), Q15_1, PPC, TD, PTA, PEE, PDK, PS, PSCBK

Figure 4.19. Regression analysis of the model.

For the purposes of analyzing the significance of each predictor in the model for hypothesis testing, the FIADEEA model utilizing the combine items from the instrument was used. Figure 4.20 shows the corresponding regression coefficient matrix.

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients		
		B	Std. Error	Beta	t	Sig.
1	(Constant)	.654	.995		.657	.512
	TD	.209	.055	.284	3.784	.000
	PC	-.291	.122	-.200	-2.383	.019
	PSCBK	-.128	.090	-.135	-1.421	.157
	PDK	.279	.091	.288	3.066	.003
	PTA	-.052	.067	-.061	-.767	.445
	PS	.065	.103	.059	.633	.528
	PPC	.195	.083	.200	2.333	.021
	PEE	.326	.113	.234	2.875	.005

a. Dependent Variable: Q13

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics				
					R Square Change	F Change	df1	df2	Sig. F Change
1	.630[a]	.397	.363	1.722	.397	11.452	8	139	.000

a. Predictors: (Constant), PC, PPC, TD, PTA, PEE, PDK, PS, PSCBK

Figure 4.20. Coefficient Matrix of the FIADEEA model with the Regression Model Summary.

The matrix shows five of the predictors were significant at the 0.05 level. Because the FIADEEA model includes a PSCBK as a predictor of TD, a Pearson's r was run to evaluate the correlation between the two predictors (Daniel 1990). The correlation was measured to be 0.289 and found to be significant at the 0.01 level.

4.3.6. Hypotheses Testing.

As mentioned in section 4.4.5, the multiple regression analysis lends support to the model by yielding significant results in 6 of the 9 predictor variables, as shown in Table 4.8. The coefficient and significance levels are denoted as (coefficient, significant value) in the Hypothesis Testing column.

Proposition	Focus	Relationship	Description	Hypotheses Testing
Proposition 0 (P0)		+	The Degree to Which Technical Debt was Considered During the Estimation Process (TD) is positively related to AEE.	Supported (.209, .000)
Proposition 1 (P1)	Project	-	The Perceived Complexity of the Effort is negatively related to AEE.	Supported (-.291, .019)
Proposition 2 (P2)	Project	-	The Perceived Scale of the Effort is negatively related to AEE.	Unsupported
Proposition 3 (P3)	Project	-	The Perceived Aggressiveness of the Effort Timeline is negatively related to AEE.	Unsupported
Proposition 4 (P4)	Team	+	The Prior Estimation Experience is positively related to AEE.	Supported (.326, .005)
Proposition 5 (P5)	Team	+	The Perceived Project Control is positively related to AEE.	Supported (.195, 0.021)
Proposition 6 (P6)	Team	+	The Problem Domain Knowledge is positively related to AEE.	Supported (.227, .003)
Proposition 7 (P7)	Team	+	The Project Specific Code Base Knowledge is positively related to TD.	Supported (.289, .000)
Proposition 8 (P8)	Team	+	The Project Specific Code Base Knowledge is positively related to AEE.	Unsupported

Table 4.8. Hypotheses testing results.

4.4. Chapter Summary.

This chapter detailed the operationalization of the model in both pilot studies and in the main study. The discussion includes the presentation of reliability measures, convergent and discriminant validity results, correlation results during the pilot study and finally multiple regression and hypothesis testing findings in the main study. Chapter V will provide details

about the FIADEEA model hypotheses as well as a discussion about the reliability and validity issues that can be improved upon in future studies.

CHAPTER V
DISCUSSION

In Chapter V, we look back to the original Chapters and revisit the original purpose of the dissertation, namely to present a new model that introduces the consideration of technical debt (TD) as a factor that influences the accuracy of effort estimates in agile projects (AEE). Following the introduction, the results of Chapter IV are discussed in conjunction with the FIADEEA model, including a discussion of the potential impact of each independent variable on the dependent variable AEE. Finally, an explanation of the reliability issues and validity issues will be provided with direction for future studies in an attempt to address the challenges encountered during the pilot and main studies.

5.1. Main Study Significance.

As detailed in Chapter I, the answer to 'How long will it take?" has been a primary goal of software development practitioners seeking to accurately estimate a software development effort. One of the first known papers addressing software effort estimation was published by Farr and Nanus in 1964 (Farr and Nanus 1964), where Farr and Nanus described the problem by saying "estimates have historically been very unreliable."

The entire software industry continues to struggle with accurately predicting the delivery dates and deliverables with sufficient accuracy. The software development world has seen a variety of software development models and corresponding estimation techniques such as

COCOMO (Boehm and Turner 2005) for more traditional software models and Planning Poker (Mahnič and Hovelja 2012) for Agile software development strategies. Practitioners continue to cite reasons for failed estimates such as aggressive timelines, scope creep, poorly documented specifications, and a poor translation of customer requirements. Academic research aimed at practitioners further backs up these common themes for inaccurate effort estimates.

In the meantime, a metaphor, technical debt, has been popularized by software development teams adopting agile software development strategies. Originally Ward Cunningham associated the term "debt" with software development in his report on the WyCash Portfolio Management System (Cunningham 1992). Today, technical debt is considered a computer programming metaphor where software developers take real or perceived implementation shortcuts during software development that eventually must be "paid" in the future. The payment is typically in the form of rewriting components of the software application. Until the payment is made in full, the impact could be any number of work-a-round or manual activities, including developing applications to monitor for errors, hiring additional employees to manually perform duties, and limiting the application functionality for customers. A more formal definition was offered by Steve McConnell, who said that technical debt is a consequence of "a *design or construction approach that's expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now (including increased cost over* time)" (Kruchten et al. 2013b).

While most agile software practitioners acknowledge the existence of technical debt, they also recognize that the ultimate solution, an accurate effort estimation, remains an elusive goal. Prior to this study, no known research has combined the two concepts into a model that may expose the relationship between technical debt and an accurate effort estimate. That is, would the

consideration of technical debt during the effort estimation process impact the accuracy of the estimate?

Overall, the significance of this problem is evident to researchers and software development practitioners alike. Even with the adoption of modern software development lifecycles and strategies, prior research shows the accuracy of effort estimates remains a challenge affecting the technology professionals worldwide. Producing academic research that advances a solution to the challenge of improving the accuracy of effort estimate is a goal of this dissertation.

5.2. The Accuracy of the Effort Estimate.

As discussed during the literature review, multiple research efforts cite the challenges of accurately predicting the time and/or effort necessary to complete a software development project or iteration. Molokken and Jorgensen reviewed the results of software survey and concluded that 60%-80% of software development estimation attempts "encounter effort and/or schedule overruns" (Molokken and Jorgensen 2003). Focusing on agile software development (ASD) efforts, Usman, Mendes, and Borstler found evidence of effort estimation challenges among agile practitioners (Usman et al. 2015). Their review of sixty (60) agile professionals revealed that "according to 52% of the respondents, effort estimation in ASD are out of range by a factor of 25% or more."

For the main study, the respondents were succinctly asked, *"For this particular iteration under consideration, how accurate was the effort estimate?"* The numeric range presented was 0-10, from 0 being **Very Late** to 10 being **On Time**. From the 150 respondents, the average value was 6.95 with a standard deviation of 2.16. Furthermore, only 17 respondents (11.33%) indicated their effort was delivered on time. The breakdown of respondents is show in Figure 5.1. This

evidence shows external validity and further echoes that ASD projects remain challenged with accurate effort estimates.

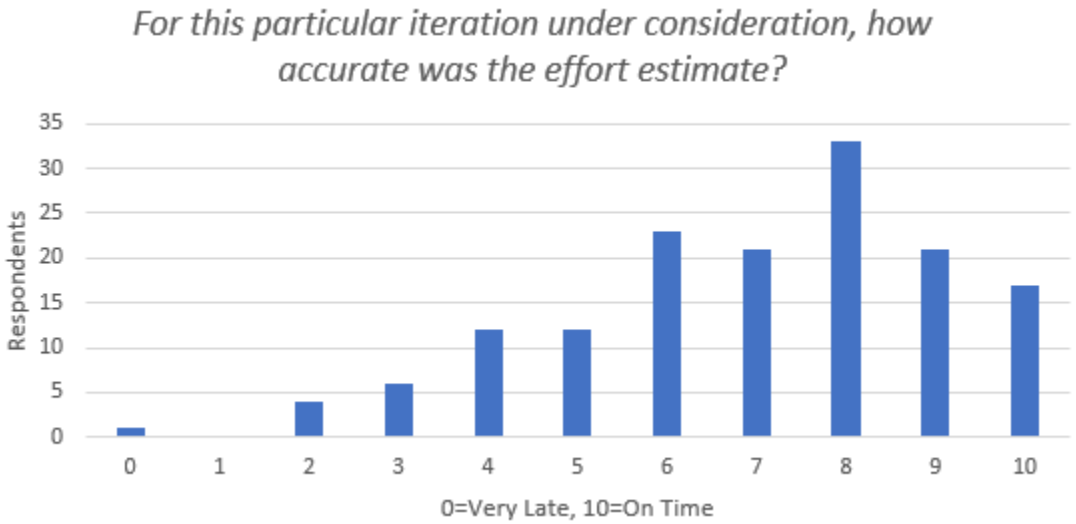


Figure 5.1. Responses to AEE instrument question.

Similar to findings during the pilot study, several respondents chose to give feedback when asked, "*Was there anything the team could have done to improve the accuracy of the effort estimate? Please explain.*" Table 5.2 highlights a variety of responses from the main study. Over 30% of respondents provided lengthy responses. For the main study, a high response rate was expected due to the design modifications to the instrument forcing a mandatory response for most items. While the narrative question remained optional, the length of the narrative responses was noteworthy and clearly required some thought on the part of the respondents. These verbose responses were helpful in explaining the practitioners' perspectives on future areas of improvement or the challenges facing the on-time delivery of their sprint or iteration requirements.

More experience people bring more accuracy
We completed the project slightly ahead of schedule and was awarded company stock as thanks! NOTE: this project spanned 24 major EDI customers, UNIX server, mainframe MVS COBOL changes and many ftp scripts.
Taken in to account external factors. The team is small and is not strictly agile in the sense that it cannot be dedicated fully to the work in the sprint.
This team was very accurate as they had been working in this domain for an average of 5+ years, had several years of Agile experience, and knew the code base very well.
Dedicating time for a research spike before beginning development on a fix for the issue would have yielded a much more accurate estimation. However, this was an escalated maintenance issue that needed to be resolved as quickly as possible. That resulted in a rough estimation beforehand.
The Development team members (Dev and QA) were based in the Ukraine and visibility into actual hours spent vs estimations is not always deemed to be accurate given I suspect that more time is sometimes being spent to complete the iteration within the sprint but not actually recorded. (this is a gut feel over a length of time working with the team and cannot be substantiated).
We are still using the point scale for stories, and not everyone has the same opinion on how to distribute the points, so that often ends up with stories of different sizes being awarded the same number of points.
There are a lot of new people on this team. It will take some time for them to be able to accurately estimate items/sprints. Also, the project is massive, and the team is on a time crunch to get everything done and delivered by certain dates for each phase. Some items were estimated extremely high for the amount of actual effort they took to develop and test.
Experience is the biggest factor (or lack thereof) of most team members when approaching a project of this scope, size and complexity. Needed more people experienced with estimation and real-world project timelines and not just the timelines we show the budget people.
This particular iteration was several iterations into the project and as a result there was good understanding of what could be accomplished. Early in the project, some of the sub-teams did well at estimation and others did not. Experience with scrum and the timeline for coding and testing improved accuracy as time went by. We had formal training early in the project, but it took several iterations to get good accuracy. This was a hard, real-time simulation for the NASA Space Station Simulator.
No, the estimate was good. The team was informed. We accepted technical debt for speed of delivery. The product owner was informed and involved. Our scope was well defined, and acceptance criteria were covering.
Team members are normally too busy working to complete sprints to do any real due diligence of tasks or the full backlog item to gain any familiarity with the code base or determine if we're actually adding more technical debt.
Considering our past experience estimating, we should increase our estimates of bugs. Particularly bugs which we are not able to reproduce or which would require above average QA efforts. Our estimates of features remains accurate.
The team in general had a pretty good feel for the estimate. However, the team was not really considered during the timeline scheduling.

Table 5.1. Narrative responses concerning areas of improvement in AEE.

5.3. FIADEEA Model with Hypotheses Results.

Given that the main study gathered evidence that challenges to AEE exist in agile software development environments, the FIADEEA model is shown in Figure 5.2 to make it easy to discuss the predictors. While identical to the conceptual model presented in previous chapters, Figure 5.2 includes the regression coefficients (see Table 4.16). Predictor variable significance is denoted as a superscript by (*) at the 0.05 level and (**) at the 0.01 level. For P7, which is a relationship between two predictor variables, the value was calculated from a Pearson's r evaluation (see Figure 5.2).

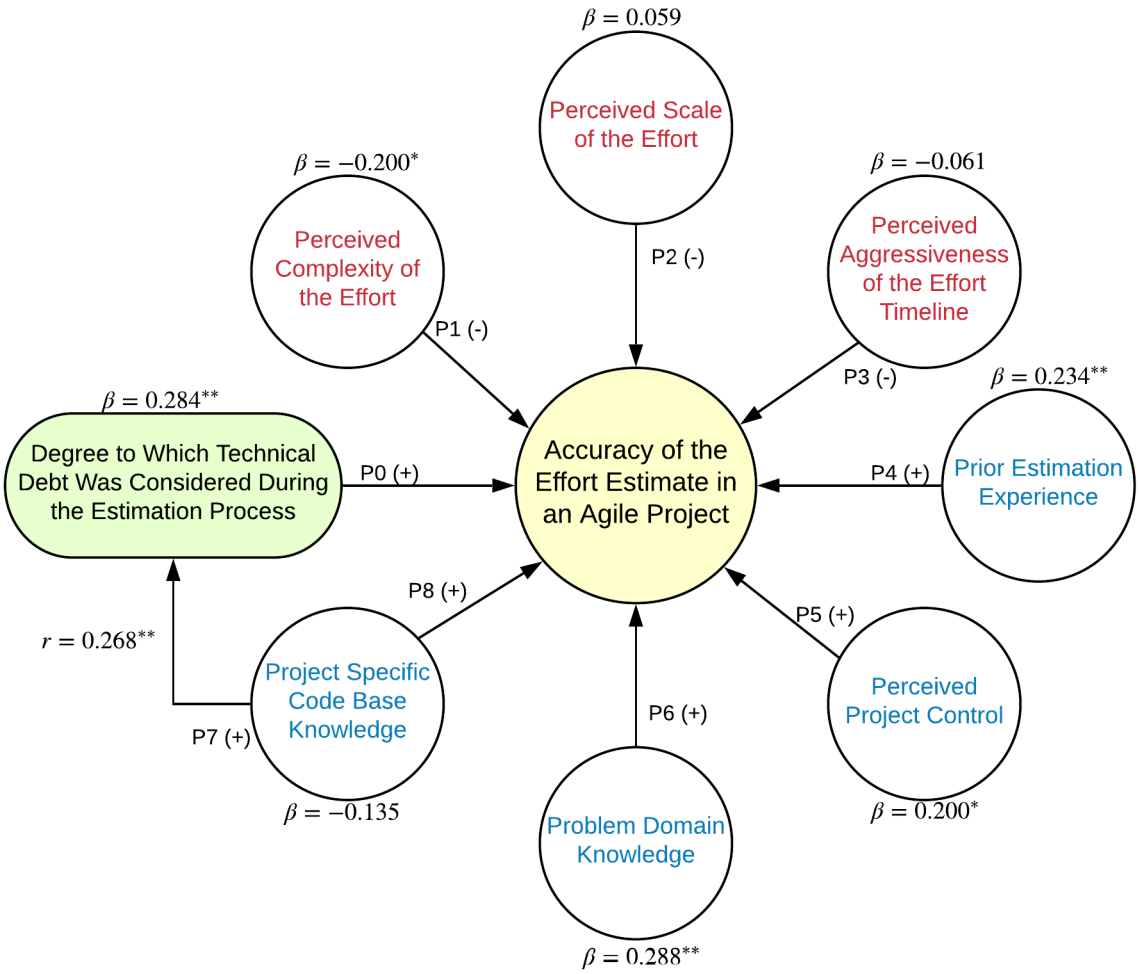


Figure 5.2. FIADEEA Model with regression standardized coefficients (betas).

As noted in Chapter IV (Figure 4.20), the regression model was significant with an R^2 value of 0.397. With R^2 ranging from 0 to 1, a desirable or high value would exceed 0.5. However, with randomized human studies, the general understanding is a R^2 exceeding 0.5 may be a challenge for inaugural research efforts. Below are hypotheses results:

1. **P0:** *The Degree to Which Technical Debt was Considered During the Estimation Process (TD) is positively related to AEE.* **There is evidence to support this proposition**. TD is shown to be significant in the regression model at the 0.01 level as shown in Table 4.8. Further, the factor analysis showed TD to be the third most important factor affecting the variance among the factors. These statistical findings support the original research question of this dissertation regarding the importance or significance of considering technical debt during the effort estimation process. These findings indicate that TD can be considered an important factor influencing the accuracy of the effort estimate (AEE).
2. **P1:** *The Perceived Complexity of the Effort is negatively related to AEE*. **There is evidence to support this proposition**. PC is shown to be significant in the regression model at the 0.05 level as shown in Table 4.8. While there were reliability concerns of PC found with our instrument, the significance finding along with the difficulty of finding a universal measure of project complexity indicates a need for further research into PC and its predictor variables.
3. **P2:** *The Perceived Scale of the Effort is negatively related to AEE.* As mentioned in the definitions section, scale is the size of the effort. **There was not statistically significant evidence to support this proposition.**

4. **P3:** *The Perceived Aggressiveness of the Effort Timeline is negatively related to AEE.* **There was not statistically significant evidence to support this proposition.** Having said that, the factor analysis indicated Perceived Timeline Aggressiveness was the second most influential factor affecting the variance from the main study data analysis. This finding warrants further isolated study of the impact of PTA on AEE.
5. **P4:** *The Prior Estimation Experience is positively related to AEE.* **There is evidence to support this proposition**. Prior Estimation Experience (PEE) as a predictor of AEE was supported and found to be significant at the 0.01 level. While it would make common sense to the practitioner that estimation experience improves the ability to accurately estimate efforts in the future, this finding indicates that the effort estimation experience of teams may improve AEE. Further, there were narrative comments from the main study that indicated the importance of seasoned or experienced team members when attempted to achieve AEE in agile iterations.
6. **P5:** *The Perceived Project Control is positively related to AEE*. **There is evidence to support this proposition**. Perceived Project Control (PPC) was found to be significant at the 0.05 level. This predictor variable indicates that practitioners whose team can influence or impact the project estimate or deliverables may improve their chances with an on-time delivery. As with PTA, this interpretation would make logical sense to the software development practitioner.
7. **P6:** *The Problem Domain Knowledge is positively related to AEE*. **There is evidence to support this proposition**. Problem Domain Knowledge (PDK) as a predictor of AEE was supported with a significance at the 0.01 level. Narrative comments tended to support that more knowledge would positively impact AEE.

8. **P7:** *The Project Specific Code Base Knowledge is positively related to TD.* **There is evidence to support this proposition.** A Pearson's r was run to evaluate the correlation between the two predictors (Daniel 1990). The correlation was measured to be 0.289 and found to be significant at the 0.01 level.
9. **P8:** *The Project Specific Code Base Knowledge is positively related to AEE*. **There was not significant evidence to support this proposition.** Upon review of the EFA results, both PDK and PSCBK loaded on the same factor, the first factor. This factor analysis finding was unexpected and somewhat contradicts the findings from Brooks who considered these items as separate factors (1983). While no conclusion can be made with the research results collected in this study, further research is warranted to determine if a second order factor representing knowledge that combines PDK, PSCBK and other knowledge factors exists.

In summary, six (6) of the nine (9) propositions shown in Table 4.8 were supported. The remaining three (3) predictors would require an exploration of alternative instruments initially before determining the best model alterations for further research and the development of an enhanced FIADEEA model.

Furthermore, returning to the standardized betas shown in Figure 5.2, a noteworthy finding is the relative significance of the standardized beta for TD at 0.284 and found to be significant at the 0.01 level (P0). The only larger standardized beta in the model was associated with PDK at 0.288 and significant at the 0.01 level (P5). However, considering the strength of the strength of the combined knowledge factors supported by a strong and significant correlation between PDK and PSCBK and that PSCBK was positively correlated to TD in the model (P7), there is evidence that TD is the most influential factor in the FIADEEA model.

5.4. Reliability Challenges.

A known benefit and challenge in the main study was related to an attempt to engage professional software practitioners across the United States. Agile software professionals are arguably some of the busiest team members in a firm or organization delivering software as a primary business objective, so the time component of a survey instrument needed to be seriously considered. In short, the availability of agile practitioners was expected to be limited so the items in the instrument would need to be completed in under ten (10) minutes.

With an evident need for timeliness, survey questions corresponding to the predictors in the model were limited to three (3). Knowing that more survey questions could lead to improved reliability (Tavakol and Dennick 2011), ideally, there would be five (5) or more questions per independent variable in the FIADEEA. But, when considering the targeted audience, the impact would potentially extend the survey by approximately five (5) minutes and could potentially reduce the participation rates.

During the pilot studies, most of the reliability concerns were addressed with the revision of instrument questions. During the main study, two (2) predictors were improved by removed a question based on the Cronbach's Alpha analysis from the SPSS results. These predictors were Project Complexity (PC) and Project Estimation Experience (PEE). However, this did not improve the reliability of PC sufficiently, as measured by a Cronbach's alpha below 0.7.

The questions used for predictors for PC were almost identical. However, one question was negatively stated, and a third question was physically grouped with other predictors. An immediate action would be to ungroup the questions in the survey instrument and allow Qualtrics to randomly present the questions to the potential respondent, which is consistent with how other

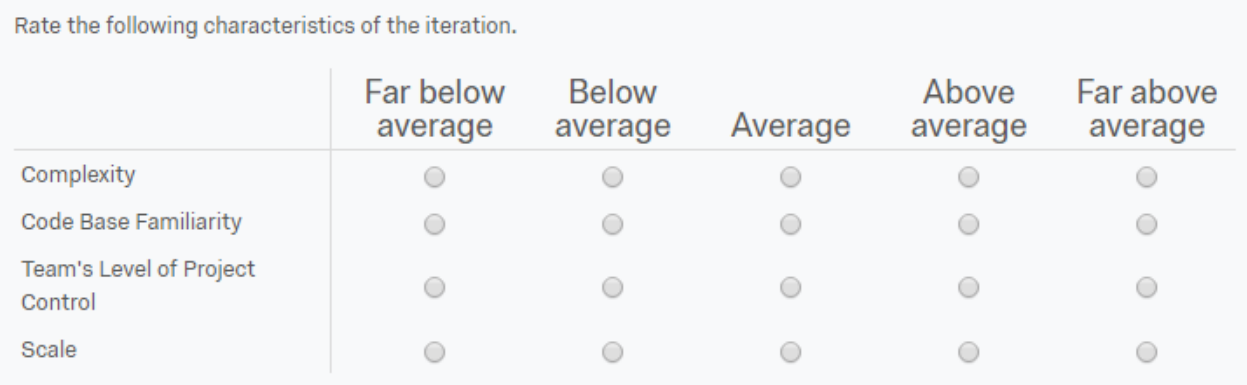
questions were randomized in the main study. However, with the PC questions being similar, further investigation may necessary to determine how complexity was interpreted across a variety of software development firms.

5.5. Validity Challenges.

A notable strength of the main study is its external validity. With multiple strategies for attracting agile software developers across the United States along with multiple screening strategies, the expectation of valid data was high.

Regarding convergent validity, the degrees of freedom dictated more responses than were available during the pilot studies. Ideally, an EFA would have been performed during the pilot studies followed by instrument revisions and a Confirmatory Factor Analysis (CFA) of the main study results. As a result, the EFA was not performed until the main study. At the end of the main study's data gathering and filtering efforts, the EFA revealed two primary issues:

1. Factor 1 loadings combined two predictors of the model, namely PDK and PSCBK.
2. Factors loadings of Question 15 items in the question group shown in Figure 5.3 were loading together.


Rate the following characteristics of the iteration.

	Far below average	Below average	Average	Above average	Far above average
Complexity	○	○	○	○	○
Code Base Familiarity	○	○	○	○	○
Team's Level of Project Control	○	○	○	○	○
Scale	○	○	○	○	○

Figure 5.3. Question 15 from the main study shows the matrix grouping.

These two issues appeared to be the primary reason the EFA loadings indicated only five (5) factors instead of the expected eight (8) factors to correspond to the eight (8) predictor variables in the FIADEEA model. Solving the second issue could be done easily in a follow-up study, where the survey instrument was modified to break up the grouping shown in Figure 5.3.

Finally, in both factor analysis efforts, the instrument items corresponding to Perceived Timeline Aggressiveness (PTA) and Consideration of Technical Debt During the Estimation Process (TD) were loading on Factor 2 and Factor 3 respectively. This gave additional confidence to this dissertation's research, exposing the potential strength of TD and its influence on the model to accurately predict the effort estimate of agile software projects.

Regarding discriminant validity, the horizontal line test against the SPSS output shown in Figure 4.13 revealed relatively few issues. The changes to the instrument outlined above to address the issues with convergent validity would also potentially address the issues with the two items not meeting the discriminant validity test.

5.6. Chapter Summary.

This chapter discussed the various benefits and challenges of conducting a main study of agile software professionals to understand the factors that influence the accuracy of an effort estimate. The study provided some support for the FIADEEA model as well as areas of potential improvement. Future research directions, the limiting conditions of this research, and the research assumptions of this study are discussed in Chapter VI.

CHAPTER VI
CONCLUSIONS

Chapter VI concludes this dissertation by first discussing the results of using the FIADEEA model to predict AEE. Following the research overview, the major contributions of this dissertation are presented, followed by the key assumptions and limitations. Finally, Chapter VI is concluded with potential future research directions.

6.1. Research Overview.

This study examined a major research question affecting software development researchers and practitioners alike, the accuracy of effort estimations. The purpose of this study was to design and propose a research model that improves the accuracy of the effort estimate in an agile software development project by considering technical debt during the effort estimation process. Stated succinctly, "*Would the consideration of technical debt during the effort estimation process in an agile software development project affect the accuracy of the estimate?"* Drawing on previous working outlined in Chapter II, the FIADEEA model was developed and presented in Chapter III. This model identified eight (8) key factors that influence the accuracy of effort estimations in agile software development projects.

Next, after IRB approval was obtained, a pilot study was performed, utilizing agile software practitioners in two software companies who were willing to assist with this research effort. Using their responses and the input from the software development team leads, a revised

pilot study was developed and targeted similar software companies willing to participate. After reliability and validity enhancements to the study were identified and implemented, the main study was performed.

The data analysis from the main study showed good reliability and validity surrounding the key factor, TD. Areas of acceptable reliability and validity were noted while further enhancements to the survey instrument were identified and discussed. The analysis results provided support for six (6) of the nine (9) proposed propositions in the FIADEEA model. Table 6.1 summarizes these results.

Proposition	Description	Testing
Proposition 0 (P0)	The Degree to Which Technical Debt was Considered During the Estimation Process (TD) is positively related to AEE.	Supported
Proposition 1 (P1)	The Perceived Complexity of the Effort is negatively related to AEE.	Supported
Proposition 2 (P2)	The Perceived Scale of the Effort is negatively related to AEE.	Unsupported
Proposition 3 (P3)	The Perceived Aggressiveness of the Effort Timeline is negatively related to AEE.	Unsupported
Proposition 4 (P4)	The Prior Estimation Experience is positively related to AEE.	Supported
Proposition 5 (P5)	The Perceived Project Control is positively related to AEE.	Supported
Proposition 6 (P6)	The Problem Domain Knowledge is positively related to AEE.	Supported
Proposition 7 (P7)	The Project Specific Code Base Knowledge is positively related to TD.	Supported
Proposition 8 (P8)	The Project Specific Code Base Knowledge is positively related to AEE.	Unsupported

Table 6.1. Testing results.

In support of the primary research question, the analysis showed TD positively affects AEE. Specifically, as Technical Debt is considered more by the agile team during the effort estimation process, the Accuracy of the Effort Estimate in an Agile Project improves and therefore

increases the likelihood for an on-time delivery. Further, factor analysis (EFA) reveals TD to be the third most important factor on AEE, in terms of variance in survey responses.

During the factor analysis, the loadings on factor 1 combined both knowledge predictors, namely the Problem Domain Knowledge (PDK) and the Project Specific Code Base Knowledge (PSCBK). While Brooks considered these separately, further research will be required to determine if a second-order knowledge factor is influencing this behavior.

6.2. Contributions.

This dissertation generated a number of theoretical and practical contributions. Below are some key contributions.

First, the study represents the first model that incorporated the consideration of technical debt into the effort estimation process for agile projects. While technical debt may be implicitly assumed by agile professionals, this study explicitly included technical debt in the model. Utilizing a study of agile software professionals provided strong external validity and exposed the positive relationship that the consideration of technical debt can have on the accuracy of the effort estimation process in an agile software project. On a more practical side, the data can guide agile training teams and companies concerning the importance of identifying and incorporating technical debt into future agile estimation efforts.

Second, the development of the FIADEEA model was the most recent development of a new model for identifying the factors which influence the accuracy of the effort estimate specific to the agile software development community. A more popular model or technique for effort estimation involves the COCOMO technique, which was primarily designed for agile alternatives, such as Waterfall.

Third, this study introduced alternative but reliable respondent selection and screening strategies in order to gather quality data on a national scale which were detailed in Chapter IV. Besides the collection of data from respondents from a variety of online agile meetup groups, the incorporation of survey research companies such as Qualtrics Research and Positly allowed for the use of mature panels that targeted software professionals.

Finally, the data collected can be used for other academic research endeavors wishing to explain or explore agile software development behaviors among US-based practitioners. Prior to this study, no similar data set of US-based agile professionals had been identified.

6.3. Assumptions and Limitations.

There are some key assumptions and limitations concerning the model and study that must be highlighted. Researchers and practitioners should be aware of these limitations when interpreting the results of this dissertation.

First, the definition of technical debt is assumed to be understandable, even among individuals who are not familiar with the agile software development methodology. The survey provided the definition of technical debt early in the instrument and prior to the key item requesting technical debt feedback.

Second, the study exposed the wide interpretation of an agile iteration, sprint or project. Unlike the waterfall strategy, agile software development strategies may be customized within an organization, which would influence the impact on the factors that influence the accuracy of the effort estimation.

Also, further studies will be necessary to determine if the FIADEEA model could sustain the rigors of becoming an internationally-supported model capable of explaining, for example,

agile models that incorporate international developers working in conjunction with US-based developers on an agile software development project.

Finally, large organizations may adapt a combination of agile and non-agile software development methodologies on large, enterprise-class, multi-year projects. Those multi-model software organizations would require further study before utilizing the FIADEEA model to analyze or correlate findings.

6.4. Future Research Directions.

Based on the results of this study, there are a number of further research directions that would benefit both researchers and practitioners. Below are a few key areas.

A simple but obvious choice would extend the study with another revised instrument in an attempt to develop a predictable strategy for agile project managers to know when technical debt and other factors were being sufficiently considered when estimating the effort.

Perceived Complexity was considered a supported predictor of AEE, but the reliability results were not sufficient to conclude that the instrument had consistently captured the meaning of complexity across respondents. Further, Perceived Scale was correlated with Perceived Complexity based on the correlation matrix shown in Figure 4.18. This may indicate the need for a separate study to explore the meaning of Perceived Complexity in conjunction with Perceived Scale within an agile software development effort.

The factor analysis finding that both PDK and PSCBK loaded on a single factor could suggest a beneficial follow-on study. While the results challenges the 1983 findings presented by Brooks, there could be a new finding specific to agile development since Brooks' findings were

focused on development prior to the adoption and popularity of agile software development strategies.

The regression model summary did not indicate that Timeline Aggressiveness (PTA) was a predictor in the model. That is, researchers cannot conclude if PTA impacts the accuracy of the effort estimate (AEE) in this study. A narrative comment suggested that some agile teams are estimating efforts independent of management setting delivery dates. As a result, a future research direction could be useful in explaining when PTA would impact AEE, specifically the management support environment for the agile team members.

Based on prior research and this dissertation, the FIADEEA model includes a number of relevant factors that may influence the accuracy of the effort estimate in agile projects. However, the FIADEEA model is just one such conceptual model. Further research could include the development of observationally equivalent models with alternative relationships which may reveal other noteworthy findings to aide in the analysis and study of AEE.

Finally, once the FIADEEA model for predicting the Accuracy of the Effort Estimate in an Agile Project has coalesced from multiple studies, tools that are currently designed to detected technical debt in a code base could be extended in an effort to develop a **FIADEEA score**, which would incorporate the detection of all FIADEEA factors solely on the codebase. This score could be a significant aid in predicting the likelihood or risk of impacting an on-time delivery based on the automated analysis of the code when observed from the beginning to the end of the sprint or iteration.

BIBLIOGRAPHY

BIBLIOGRAPHY

Acuna, S. T., Juristo, N., and Moreno, A. M. 2006. "Emphasizing Human Capabilities in Software Development," *IEEE Software* (23:2), pp. 94–101. (https://doi.org/10.1109/MS.2006.47).

Agile Is the New Normal. 2017. Hewlett Packard Enterprise.

Alves, N. S. R., Mendes, T. S., de Mendonça, M. G., Spínola, R. O., Shull, F., and Seaman, C. 2016. "Identification and Management of Technical Debt: A Systematic Mapping Study," *Information and Software Technology* (70), pp. 100–121. (https://doi.org/10.1016/j.infsof.2015.10.008).

Avgeriou, P., Kruchten, P., Nord, R. L., Ozkaya, I., and Seaman, C. 2016. "Reducing Friction in Software Development," *IEEE Software* (33:1), pp. 66–73. (https://doi.org/10.1109/MS.2016.13).

Banker, R. D., Davis, G. B., and Slaughter, S. A. 1998. "Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study," *Management Science* (44:4), pp. 433–450.

Boehm, B., and Turner, R. 2005. "Management Challenges to Implementing Agile Processes in Traditional Development Organizations," *IEEE Software* (22:5), pp. 30–39. (https://doi.org/10.1109/MS.2005.129).

Boehm, B. W. 1981. *Software Engineering Economics*, (1 edition.), Englewood Cliffs, N.J: Prentice Hall.

Brooks, R. 1983. "Towards a Theory of the Comprehension of Computer Programs," *International Journal of Man-Machine Studies* (18:6), pp. 543–554. (https://doi.org/10.1016/S0020-7373(83)80031-5).

Cattell, R. B., Eber, H. W., and Tatsuoka, M. M. 1970. *Handbook for the Sixteen Personality Factor Questionnaire (16 PF): In Clinical, Educational, Industrial, and Research Psychology, for Use with All Forms of the Test,* Institute for Personality and Ability Testing.

Churchill, G. A. 1979. "A Paradigm for Developing Better Measures of Marketing Constructs," *Journal of Marketing Research* (16:1), pp. 64–73. (https://doi.org/10.2307/3150876).

Clancy, T. 1995. "Chaos Report," The Standish Group Report.

Clark, B., Devnani-Chulani, S., and Boehm, B. 1998. "Calibrating the COCOMO II Post-Architecture Model," in *Proceedings of the 20th International Conference on Software Engineering*, , April, pp. 477–480. (https://doi.org/10.1109/ICSE.1998.671610).

"COCOMO." 2017. *Wikipedia*. (https://en.wikipedia.org/w/index.php?title=COCOMO&oldid=768471969).

Coelho, E., and Basu, A. 2012. "Effort Estimation in Agile Software Development Using Story Points," *International Journal of Applied Information Systems (IJAIS)* (3:7). (https://pdfs.semanticscholar.org/0ae7/5cc0678f6b01f1066093b19cad9dd4379385.pdf).

Cunningham, W. 1992. "The WyCash Portfolio Management System," in *Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications (Addendum)*, OOPSLA '92, New York, NY, USA: ACM, pp. 29–30. (https://doi.org/10.1145/157709.157715).

Daniel, W. W. 1990. *Applied Nonparametric Statistics*, PWS-Kent Publ.

Dikert, K., Paasivaara, M., and Lassenius, C. 2016. "Challenges and Success Factors for Large-Scale Agile Transformations: A Systematic Literature Review," *Journal of Systems and Software* (119), pp. 87–108. (https://doi.org/10.1016/j.jss.2016.06.013).

Dingsøyr, T., and Lassenius, C. 2016. "Emerging Themes in Agile Software Development: Introduction to the Special Section on Continuous Value Delivery," *Information and Software Technology* (77), pp. 56–60. (https://doi.org/10.1016/j.infsof.2016.04.018).

Dybå, T., and Dingsøyr, T. 2008. "Empirical Studies of Agile Software Development: A Systematic Review," *Information and Software Technology* (50:9), pp. 833–859. (https://doi.org/10.1016/j.infsof.2008.01.006).

Farr, L., and Nanus, B. 1964. "Factors That Affect the Cost of Computer Programming," SYSTEM DEVELOPMENT CORP SANTA MONICA CA.

Flyvbjerg, B., and Budzier, A. 2011. "Why Your IT Project May Be Riskier Than You Think," *Harvard Business Review*, , September 1. (https://hbr.org/2011/09/why-your-it-project-may-be-riskier-than-you-think, accessed March 12, 2018).

Jarvis, C. B., MacKenzie, S. B., and Podsakoff, P. M. 2003. "A Critical Review of Construct Indicators and Measurement Model Misspecification in Marketing and Consumer Research," *Journal of Consumer Research* (30:2), pp. 199–218.

Kazman, R., Cai, Y., Mo, R., Feng, Q., Xiao, L., Haziyev, S., Fedak, V., and Shapochka, A. 2015. "A Case Study in Locating the Architectural Roots of Technical Debt," in *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ICSE '15, Piscataway, NJ, USA: IEEE Press, pp. 179–188. (http://dl.acm.org/citation.cfm?id=2819009.2819037).

Kruchten, P., Nord, R. L., and Ozkaya, I. 2012. “Technical Debt: From Metaphor to Theory and Practice.,” *IEEE Software* (29:6). (http://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=07407459&AN=83329085&h=BURWY576KTQh8AGUleFKxq95IfhLCKyly3vdjp0cDMS5eD5d59lo3pw6a0lpv16TOEOWEBKZ2Tqhm%2BGUN9ToCg%3D%3D&crl=c).

Kruchten, P., Nord, R. L., Ozkaya, I., and Falessi, D. 2013a. “Technical Debt: Towards a Crisper Definition Report on the 4th International Workshop on Managing Technical Debt,” *SIGSOFT Softw. Eng. Notes* (38:5), pp. 51–54. (https://doi.org/10.1145/2507288.2507326).

Kruchten, P., Nord, R. L., Ozkaya, I., and Falessi, D. 2013b. “Technical Debt: Towards a Crisper Definition Report on the 4th International Workshop on Managing Technical Debt,” *ACM SIGSOFT Software Engineering Notes* (38:5), p. 51. (https://doi.org/10.1145/2507288.2507326).

Laqrichi, S., Gourc, D., and Marmier, F. 2015. “Toward an Effort Estimation Model for Software Projects Integrating Risk,” *ArXiv Preprint ArXiv:1509.00602*. (https://arxiv.org/abs/1509.00602).

Lim, E., Taksande, N., and Seaman, C. 2012. “A Balancing Act: What Software Practitioners Have to Say about Technical Debt,” *IEEE Software* (29:6), pp. 22–27. (https://doi.org/10.1109/MS.2012.130).

Mahnič, V., and Hovelja, T. 2012. “On Using Planning Poker for Estimating User Stories,” *Journal of Systems and Software* (85:9), Selected Papers from the 2011 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2011), pp. 2086–2095. (https://doi.org/10.1016/j.jss.2012.04.005).

“Manifesto for Agile Software Development.” 2001. , February. (http://agilemanifesto.org/, accessed October 21, 2017).

Mäntylä, M. V., and Lassenius, C. 2006. “Drivers for Software Refactoring Decisions,” in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, ACM, pp. 297–306.

McCullagh, P. 1980. “Regression Models for Ordinal Data,” *Journal of the Royal Statistical Society. Series B (Methodological)* (42:2), pp. 109–142.

Molokken, K., and Jorgensen, M. 2003. “A Review of Software Surveys on Software Effort Estimation,” in *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.*, , September, pp. 223–230. (https://doi.org/10.1109/ISESE.2003.1237981).

Morgenthaler, J. D., Gridnev, M., Sauciuc, R., and Bhansali, S. 2012. “Searching for Build Debt: Experiences Managing Technical Debt at Google,” in *Proceedings of the Third*

International Workshop on Managing Technical Debt, MTD '12, Piscataway, NJ, USA: IEEE Press, pp. 1–6. (http://dl.acm.org/citation.cfm?id=2666036.2666037).

Myers, M. D. 2008. *Qualitative Research in Business & Management*, SAGE.

Nasser, V. H. (n.d.). *What Factors Impact Effort Estimation Accuracy?*

Nguyen-Cong, D., and Tran-Cao, D. 2013. "A Review of Effort Estimation Studies in Agile, Iterative and Incremental Software Development," in *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2013 IEEE RIVF International Conference On*, IEEE, pp. 27–30. (http://ieeexplore.ieee.org/abstract/document/6719861/).

Petscher, Y., Schatschneider, C., and Compton, D. L. 2013. *Applied Quantitative Analysis in Education and the Social Sciences*, Routledge.

"PlanningPoker.Com - Estimates Made Easy. Sprints Made Simple." (n.d.). *PlanningPoker.Com*. (https://www.planningpoker.com/, accessed April 29, 2017).

"Positly: Study Recruiter." (n.d.). *Positly: Study Recruiter*. (https://www.positly.com/about/, accessed February 24, 2019).

Qurashi, S. A., and Qureshi, M. 2014. "Scrum of Scrums Solution for Large Size Teams Using Scrum Methodology," *ArXiv Preprint ArXiv:1408.6142*. (https://arxiv.org/abs/1408.6142).

Reithel, B. 2009. *Conceptual Model Development Process - An Iterative Approach*, presented at the MIS 665 Lecture, Spring 2009, , January.

Roulin, N. 2015. "Don't Throw the Baby Out With the Bathwater: Comparing Data Quality of Crowdsourcing, Online Panels, and Student Samples," *Industrial and Organizational Psychology* (8:2), pp. 190–196. (https://doi.org/10.1017/iop.2015.24).

Ruparelia, N. B. 2010. "Software Development Lifecycle Models," *ACM SIGSOFT Software Engineering Notes* (35:3), p. 8. (https://doi.org/10.1145/1764810.1764814).

"Software Developers, Applications & Systems Software." (n.d.). *Data USA*. (https://datausa.io/profile/soc/15113X/#demographics, accessed March 2, 2019).

"Software Development Trends 2018: Latest Research and Data." (n.d.). *Coding Sans*. (https://codingsans.com/blog/software-development-trends-2018, accessed April 11, 2018).

Tavakol, M., and Dennick, R. 2011. "Making Sense of Cronbach's Alpha," *International Journal of Medical Education* (2), pp. 53–55. (https://doi.org/10.5116/ijme.4dfb.8dfd).

"The Leading Research & Experience Software." (n.d.). *Qualtrics*. (https://www.qualtrics.com/, accessed July 1, 2018).

Turk, D., France, R., and Rumpe, B. 2014. "Assumptions Underlying Agile Software Development Processes," *ArXiv Preprint ArXiv:1409.6610*. (https://arxiv.org/abs/1409.6610).

Usman, M., Mendes, E., and Börstler, J. 2015. *Effort Estimation in Agile Software Development: A Survey on the State of the Practice*, presented at the Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, ACM, April 27, p. 12. (https://doi.org/10.1145/2745802.2745813).

Usman, M., Mendes, E., Weidt, F., and Britto, R. 2014. *Effort Estimation in Agile Software Development: A Systematic Literature Review*, ACM Press, pp. 82–91. (https://doi.org/10.1145/2639490.2639503).

Vijayasarathy, L. R., and Butler, C. W. 2016. "Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter?," *IEEE Software* (33:5), pp. 86–94. (https://doi.org/10.1109/MS.2015.26).

Weaver, B., and Wuensch, K. L. 2013. "SPSS and SAS Programs for Comparing Pearson Correlations and OLS Regression Coefficients," *Behavior Research Methods* (45:3), pp. 880–895. (https://doi.org/10.3758/s13428-012-0289-7).

Whitworth, E., and Biddle, R. 2007. "The Social Nature of Agile Teams," in *Agile Conference (AGILE), 2007*, IEEE, pp. 26–36.

Williams, M. 2002. *Microsoft Visual C# (Core Reference)*, Redmond, WA, USA: Microsoft Press.

Xiao, L., Cai, Y., and Kazman, R. 2014. *Titan: A Toolset That Connects Software Architecture with Quality Analysis*, ACM Press, pp. 763–766. (https://doi.org/10.1145/2635868.2661677).

Ziauddin, S. K. T., and Zia, S. 2012. "An Effort Estimation Model for Agile Software Development," *Advances in Computer Science and Its Applications (ACSA)* (314), pp. 314–324.

LIST OF APPENDICES

APPENDIX A: INITIAL PILOT STUDY INSTRUMENT

Thank you for taking this brief survey. The goal of this research is to evaluate the influence of various factors on the accuracy of effort estimation in an agile software project.

Estimated time of completion is approximately five (5) minutes. Individuals must be 18 years of age to participate. Below are brief survey definitions that will be used throughout the survey. Please take a moment to become familiar with these definitions.

Your participation is voluntary. Your responses will remain anonymous. You may halt your participation in this study at any time. Further, you may skip any question that you do not wish to answer.

IRB Approval

This study has been reviewed by The University of Mississippi's Institutional Review Board (IRB). If you have any questions, concerns, or reports regarding your rights as a participant of research, please contact the IRB at (662) 915-7482 or irb@olemiss.edu. ***Statement of Consent*** *I have read and understand the above information. By completing the survey/interview I consent to participate in the study.*

Survey Definitions

Effort Estimation: Attempting to quantify the work or time required to complete a task. An accurate response to "When will you be done?" is a primary goal of effort estimation.
Iteration: In agile software development, a single development cycle.
Software Methodology: A strategy for developing software that includes software development, testing, and delivery. Commonly discussed software methodologies include agile, iterative, and waterfall.
Sprint: The name use for an iteration specific to Scrum agile methods.
Technical Debt: Steve McConnell succinctly described technical debt as "a design or construction approach that's expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now (including increased cost over time)." In short, technical debt is a consequence of a previous development decision rather than a software bug or defect.

☐ By selecting this checkbox, I certify that I am 18 years or older. (1)

Q2 Within your organization, which title describes your current role?

- ○ Executive (1)
- ○ Developer (2)
- ○ Personnel Manager (3)
- ○ Product Manager (4)
- ○ Project Manager (5)
- ○ Tester (6)

Q3 Which answer best describes your primary type of customer?

- ○ Internal unit (1)
- ○ Individual consumers (2)
- ○ Business or firm (3)
- ○ Open source community (4)
- ○ All of the above (5)

Q4 What is your age?

- ○ 18-24 (1)
- ○ 25-34 (2)
- ○ 35-44 (3)
- ○ 45-54 (4)
- ○ 55-64 (5)
- ○ 65+ (6)

Q5 What is your gender identity?

- ○ Female (1)
- ○ Male (2)
- ○ Other (3) _____
- ○ I prefer not to answer (4)

Q6 How many years of experience do you have with effort estimation in software development?

- ○ 0-4 (1)
- ○ 5-9 (2)
- ○ 10-14 (3)
- ○ 15-19 (4)
- ○ 20+ (5)

Q7 How many years of experience do you have working with agile software development teams?

- ○ 0-4 (1)
- ○ 5-9 (2)
- ○ 10-14 (3)
- ○ 15-19 (4)
- ○ 20+ (5)

Q8 Which agile software development methods or strategies have you used? Check all that apply.

- ☐ Adaptive software development (1)
- ☐ Crystal (2)
- ☐ Dynamic systems development (3)
- ☐ Extreme programming (4)
- ☐ Feature-driven development (5)
- ☐ Lean software development (6)
- ☐ Rapid application development (7)
- ☐ Scrum (8)
- ☐ Other (9) _____

Q9 Are you familiar with technical debt in a software development environment?

- ○ I have never heard of it (1)
- ○ I have heard of it, but I don't know what it means (2)
- ○ I have a general understanding of it (3)
- ○ I have a understanding of it and have actively considered it in the past (4)
- ○ I have extensive knowledge of it and take it into consideration frequently (5)

Q10 Which tools or techniques have you used to assist with an agile estimation effort? Check all that apply.

- ☐ Affinity Mapping (1)
- ☐ Bucket system (2)
- ☐ Dot Voting (3)
- ☐ Expert opinion (SWAG) (4)
- ☐ Historical data (5)
- ☐ Ordering Protocol (6)
- ☐ Planning poker (7)
- ☐ Relative mass valuation (8)
- ☐ T-Shirt Sizes (9)
- ☐ Other (10) _____

Q11 **Instructions:** For the remaining survey questions, please recall your experience as a team participant during a completed iteration or sprint that involved the maintenance of an existing piece of code. Using that experience, provide the best answer.

Q12 In order to group responses pertaining to the same iteration or sprint, please provide the first and last initial of the agile team lead followed by the end date ("MMDDYYYY") of the iteration or sprint (e.g., TC04162018).

_____

Q13 For the particular iteration under consideration, how accurate was the effort estimation?

	Not Accurate									Very Accurate
	0	1	2	3	4	5	6	7	8	9	10
Accuracy ()											

Q14 Rate the team's overall skill level for these tasks.

- ○ Needs significant improvement (1)
- ○ Needs improvement (2)
- ○ Barely adequate skills (3)
- ○ Capable and effective (4)
- ○ Very capable/ideal (5)

Q15 Rate the following components of your iteration.

	Far below average (1)	Somewhat below average (2)	Average (3)	Somewhat above average (4)	Far above average (5)
Level of Difficulty (1)	○	○	○	○	○
Timeline Aggresiveness (2)	○	○	○	○	○
Deadline Flexibility (3)	○	○	○	○	○
Code Base Familiarity (4)	○	○	○	○	○

Q16 How many employees were assigned to this iteration?

- ○ 1-4 (1)
- ○ 5-9 (2)
- ○ 10-24 (3)
- ○ 25-49 (4)
- ○ 50-99 (5)
- ○ 100+ (6)

Q17 Did management pressure your team to work extended hours to complete the effort on schedule?

- ○ None at all (1)
- ○ A little (2)
- ○ Somewhat (3)
- ○ A lot (4)
- ○ A whole lot (5)

Q18 Was the project leadership effective in managing customer expectations?

- ○ Not effective at all (1)
- ○ Barely effective (2)
- ○ Moderately effective (3)
- ○ Very effective (4)
- ○ Completely effective (5)

Q19 Was a financial incentive suggested or committed for the on-time delivery of this iteration or project effort?

- ○ Definitely not (1)
- ○ I don't think so (2)
- ○ I don't know (3)
- ○ I think so (4)
- ○ Definitely yes (5)

Q20 How were the agile team members geographically dispersed?

- ○ 1 location (1)
- ○ 2 or more locations in the same city (2)
- ○ 2 or more locations in the same time zone (3)
- ○ 2 or more locations across time zones but in the same country (4)
- ○ Internationally dispersed (5)

Q21 When determining the specific deliverables for your effort, was your input requested and taken into consideration?

- ○ Not at all (1)
- ○ Input was requested but not considered (2)
- ○ Input was requested and partially considered (3)
- ○ Input was requested and given full consideration (4)
- ○ Input was requested, given full consideration and implemented (5)

Q22 When addressing the tasks in this iteration or sprint, rate your satisfaction with the programming techniques and strategies utilized by the team.

- ○ Not satisfied at all (1)
- ○ Somewhat dissatisfied (2)
- ○ Somewhat satisfied (3)
- ○ Very satisfied (4)
- ○ Completed satisfied (5)

Q23 What was the length of the project associated with this iteration?

- ○ Up to 2 weeks (1)
- ○ 3 - 4 weeks (2)
- ○ 5 - 8 weeks (3)
- ○ 9 - 12 weeks (4)
- ○ 13- 16 weeks (5)
- ○ 17 - 52 weeks (6)
- ○ More than 1 year (7)

Q24 What was the team's level of familiarity with the **business** problem associated with this iteration?

- ○ Not familiar at all (1)
- ○ Barely familiar (2)
- ○ Moderately familiar (3)
- ○ Very familiar (4)
- ○ Completely familiar (5)

Q25 To complete this iteration, rate the level of sophistication needed to complete the tasks?

- ○ Not sophisticated at all (e.g., fundamental programming techniques) (1)
- ○ Barely sophisticated (e.g., modular development) (2)
- ○ Moderately sophisticated (e.g., integration with existing components and interfaces) (3)
- ○ Sophisticated (e.g., third party integrations, performance sensitive routines) (4)
- ○ Very Sophisticated (e.g., complex algorithms, complex data structures) (5)

Q26 Was technical debt considered during the effort estimation process?

As a reminder of the definition of technical debt, Steve McConnell succinctly described technical debt as "a design or construction approach that's expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now (including increased cost over time)." In short, technical debt is a consequence of a previous development decision rather than a software bug or defect.

- O Not at all (1)
- O A little (2)
- O Somewhat (3)
- O A lot (4)
- O A great deal (5)

Q27 After the iteration, were there any unexpected application behaviors or significant issues arising from the complexity of the iteration requirements?

- ○ A significant number of issues came up (1)
- ○ We had a few significant issues (2)
- ○ We had a few moderate issues (3)
- ○ We had a few minor issues (4)
- ○ We had no issues (5)

Q28 Did your agile team lead (e.g., Scrum master, etc.) discuss technical debt prevention during your team meetings?

- ○ None at all (1)
- ○ A little (2)
- ○ Somewhat (3)
- ○ A lot (4)
- ○ A great deal (5)

Q29 Was the team familiar with the code base during this iteration?

- ○ Not familiar at all (1)
- ○ Barely familiar (2)
- ○ Moderately familiar (3)
- ○ Very familiar (4)
- ○ Completely familiar (5)

Q30 During the post iteration review, did you team uncover any issues stemming from the complexity of the iteration requirements.

- ○ We had several issues (1)
- ○ We had a few significant issues (2)
- ○ We had a few moderate issues (3)
- ○ We had a few minor issues (4)
- ○ We had no issues (5)

Q31 Was there anything the team could have done to improve the accuracy of the effort estimate? Please explain.

_____

_____

Q32 Thank you for participating in this study.

(Optional) If your team would like to see the final results of this study, please leave an email address where the results can be sent to you.

APPENDIX B: REVISED PILOT STUDY INSTRUMENT

Thank you for taking this brief survey. The goal of this research is to evaluate the influence of various factors on the accuracy of effort estimation in an agile software project.

Estimated time of completion is approximately five (5) minutes. Individuals must be 18 years of age to participate. Below are brief survey definitions that will be used throughout the survey. Please take a moment to become familiar with these definitions.

Your participation is voluntary. Your responses will remain anonymous. You may halt your participation in this study at any time. Further, you may skip any question that you do not wish to answer.

IRB Approval

This study has been reviewed by The University of Mississippi's Institutional Review Board (IRB). If you have any questions, concerns, or reports regarding your rights as a participant of research, please contact the IRB at (662) 915-7482 or irb@olemiss.edu. ***Statement of Consent*** *I have read and understand the above information. By completing the survey/interview I consent to participate in the study.*

Survey Definitions

Effort Estimation: Attempting to quantify the work or time required to complete a task. An accurate response to "When will you be done?" is a primary goal of effort estimation.
Iteration: In agile software development, a single development cycle.
Software Methodology: A strategy for developing software that includes software development, testing, and delivery. Commonly discussed software methodologies include agile, iterative, and waterfall.
Sprint: The name use for an iteration specific to Scrum agile methods.
Technical Debt: Steve McConnell succinctly described technical debt as "a design or construction approach that's expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now (including increased cost over time)." In short, technical debt is a consequence of a previous development decision rather than a software bug or defect.

☐ By selecting this checkbox, I certify that I am 18 years or older. (1)

Q2 Within your organization, which title describes your current role?

- ○ Executive (1)
- ○ Developer (2)
- ○ Personnel Manager (3)
- ○ Product Manager (4)
- ○ Project Manager (5)
- ○ Tester (6)

Q3 Which answer best describes your primary type of customer?

- ○ Internal unit (1)
- ○ Individual consumers (2)
- ○ Business or firm (3)
- ○ Open source community (4)
- ○ All of the above (5)

Q4 What is your age?

- ○ 18-24 (1)
- ○ 25-34 (2)
- ○ 35-44 (3)
- ○ 45-54 (4)
- ○ 55-64 (5)
- ○ 65+ (6)

Q5 What is your gender identity?

- ○ Female (1)
- ○ Male (2)
- ○ Other (3) _____
- ○ I prefer not to answer (4)

Q11 **Instructions:** For the remaining survey questions, please recall your experience as a team participant during a completed iteration or sprint that involved the maintenance of an existing piece of code. Using that experience, provide the best answer.

Q12 In order to group responses pertaining to the same iteration or sprint, please provide the first and last initial of the agile team lead followed by the end date ("MMDDYYYY") of the iteration or sprint (e.g., TC04162018).

_____

Q7 Estimate the average years of experience your team members have with software development effort estimates.

- ○ 0-4 years (1)
- ○ 5-9 years (2)
- ○ 10-14 years (3)
- ○ 15-19 years (4)
- ○ 20+ years (5)

Q34 Rate your team's effort estimation experience.

- None at all (1)
- A little (2)
- Somewhat (3)
- A lot (4)
- A whole lot (5)

Q36 Was the team's effort estimation experience suitable for this project?

- Not at all suitable (1)
- Slightly suitable (2)
- Somewhat suitable (3)
- Moderately suitable (4)
- Very Suitable (5)

Q13 For the particular iteration under consideration, how accurate was the effort estimation?

Q18 Was the team effective in controlling the project?

- ○ Not effective at all (1)
- ○ Barely effective (2)
- ○ Moderately effective (3)
- ○ Very effective (4)
- ○ Completely effective (5)

Q21 Was the team permitted to control the project?

- ○ Not at all (1)
- ○ Input was requested but not considered (2)
- ○ Input was requested and partially considered (3)
- ○ Input was requested and given full consideration (4)
- ○ Input was requested, given full consideration and implemented (5)

Q15 Rate the following components of your iteration.

	Far below average (1)	Somewhat below average (2)	Average (3)	Somewhat above average (4)	Far above average (5)
Timeline Aggresiveness (2)	○	○	○	○	○
Project Control (3)	○	○	○	○	○
Code Base Familiarity (4)	○	○	○	○	○
Effort Complexity (5)	○	○	○	○	○
Project Scale (6)	○	○	○	○	○

Q22 When addressing the tasks for this iteration or sprint, how familiar were the team members with the code base for this project?

- ○ Not at all familiar (1)
- ○ Slightly familiar (2)
- ○ Somewhat familiar (3)
- ○ Moderately familiar (4)
- ○ Extremely familiar (5)

Q29 Rate the team's knowledge about the specific code base being modified during this iteration.

- ○ Not understandable at all (1)
- ○ Barely understandable (2)
- ○ Moderately understandable (3)
- ○ Very understandable (4)
- ○ Completely understandable (5)

Q17 How would your team rate the aggressiveness of the iteration timeline?

- ○ Not aggressive at all (1)
- ○ Slightly aggressive (2)
- ○ Moderately aggressive (3)
- ○ Very aggressive (4)
- ○ Extremely aggressive (5)

Q19 Indicate your level of agreement with the following statement:

Overall, the team felt the timeline for this iteration was aggressive.

- ○ Strongly disagree (1)
- ○ Disagree (2)
- ○ Neither agree or disagree (3)
- ○ Agree (4)
- ○ Strongly agree (5)

Q16 How would the team rate the scale of this project?

- ○ Very small (1)
- ○ Small (2)
- ○ Average (3)
- ○ Large (4)
- ○ Very Large (5)

Q23 What was the team's perception of the scale of the project?

- ○ Very small (1)
- ○ Small (2)
- ○ Average (3)
- ○ Large (4)
- ○ Very Large (5)

Q14 Rate the team's overall knowledge of the problem domain associated with this iteration.

- ○ Not at all knowledgeable (1)
- ○ Slightly knowledgeable (2)
- ○ Somewhat knowledgeable (3)
- ○ Moderately knowledgeable (4)
- ○ Extremely knowledgeable (5)

Q24 What was the team's knowledge of the business issues associated with this iteration?

- ○ Not at all knowledgeable (1)
- ○ Slightly knowledgeable (2)
- ○ Somewhat knowledgeable (3)
- ○ Moderately knowledgeable (4)
- ○ Extremely knowledgeable (5)

Q25 Indicate your level of agreement with the following statement:

For this iteration, the team's knowledge of the business problem domain was adequate.

- ☐ Strongly disagree (1)
- ☐ Disagree (2)
- ☐ Neither agree or disagree (3)
- ☐ Agree (4)
- ☐ Strongly agree (5)

Q26 Was technical debt considered during the effort estimation process?
As a reminder of the definition of technical debt, Steve McConnell succinctly described technical debt as "a design or construction approach that's expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now (including increased cost over time)." In short, technical debt is a consequence of a previous development decision rather than a software bug or defect.

- ○ Not at all (1)
- ○ A little (2)
- ○ Somewhat (3)
- ○ A lot (4)
- ○ A great deal (5)

Q33 During any team meetings about effort estimation, did agile team members discuss technical debt?

- ○ Not at all (1)
- ○ A little (2)
- ○ Somewhat (3)
- ○ A lot (4)
- ○ A great deal (5)

Q28 Did your agile team lead (e.g., Scrum master, etc.) allow discussions of technical debt during the effort estimation process?

- ○ None at all (1)
- ○ A little (2)
- ○ Somewhat (3)
- ○ A lot (4)
- ○ A great deal (5)

Q27 Rate the team's perception of the complexity of this effort.

- ○ Very complex (1)
- ○ Moderately complex (2)
- ○ Somewhat complex (3)
- ○ Slightly complex (4)
- ○ Not complex (5)

Q30 Indicate your level of agreement with the following statement:

The complexity of this iteration was excessive.

- ○ Strongly disagree (1)
- ○ Disagree (2)
- ○ Neither agree or disagree (3)
- ○ Agree (4)
- ○ Strongly agree (5)

Q31 Was there anything the team could have done to improve the accuracy of the effort estimate? Please explain.

_____

Q32 Thank you for participating in this study.

(Optional) If your team would like to see the final results of this study, please leave an email address where the results can be sent to you.

APPENDIX C: FINAL PILOT STUDY INSTRUMENT

Thank you for taking this brief survey. The goal of this research is to evaluate the influence of various factors on the accuracy of effort estimation in an agile software project.

Estimated time of completion is approximately five (5) minutes. Individuals must be 18 years of age to participate. Below are brief survey definitions that will be used throughout the survey. Please take a moment to become familiar with these definitions.

Your participation is voluntary. Your responses will remain anonymous. You may halt your participation in this study at any time. Further, you may skip any question that you do not wish to answer.

IRB Approval

This study has been reviewed by The University of Mississippi's Institutional Review Board (IRB). If you have any questions, concerns, or reports regarding your rights as a participant of research, please contact the IRB at (662) 915-7482 or irb@olemiss.edu. ***Statement of Consent*** *I have read and understand the above information. By completing the survey/interview I consent to participate in the study.*

Survey Definitions

Effort Estimation: Attempting to quantify the work or time required to complete a task. An accurate response to "When will you be done?" is a primary goal of effort estimation.
Iteration: In agile software development, a single development cycle.
Software Methodology: A strategy for developing software that includes software development, testing, and delivery. Commonly discussed software methodologies include agile, iterative, and waterfall.
Sprint: The name used for an iteration specific to Scrum agile methods.
Technical Debt: Steve McConnell succinctly described technical debt as a consequence of "a design or construction approach that's expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now (including increased cost over time)." In short, technical debt is a consequence of a previous development decision rather than a software bug or defect.

☐ By selecting this checkbox, I certify that I am 18 years or older. (1)

Q2 Within your organization, which title describes your current role?

- ○ Executive (1)
- ○ Developer (2)
- ○ Personnel Manager (3)
- ○ Product Manager (4)
- ○ Project Manager (5)
- ○ Tester (6)

Q4 What is your age?

_____

Q32 How many years of software development experience do you have?

_____

Q5 What is your gender identity?

- ○ Female (1)
- ○ Male (2)
- ○ Other (3) _____
- ○ I prefer not to answer (4)

Q11 **Instructions:** For the remaining survey questions, please recall your experience as a team participant during a completed iteration or sprint that involved the maintenance of an existing piece of code. Using that experience, provide the best answer.

Q12 In order to group responses pertaining to the same iteration or sprint, please provide the first and last initial of the agile team lead followed by the end date ("MMDDYYYY") of the iteration or sprint (e.g., TC04162018).

_____

Q34 Related to this iteration or sprint, what is the expected useful lifetime of the software (in years)?

Q7 Indicate your level of agreement with the following statement.

The team had limited effort estimation experience.

- ○ Strongly disagree (1)
- ○ Disagree (2)
- ○ Neither agree or disagree (3)
- ○ Agree (4)
- ○ Strongly agree (5)

Q34 Rate your team's effort estimation experience.

- None at all (1)
- A little (2)
- Somewhat (3)
- A lot (4)
- A whole lot (5)

Q36 Indicate your level of agreement with the following statement.

The team had a fair amount of effort estimation experience.

- Strongly disagree (1)
- Disagree (2)
- Neither agree or disagree (3)
- Agree (4)
- Strongly agree (5)

Q13 For the particular iteration under consideration, how accurate was the effort estimate?

Q18 Indicate your level of agreement with the following statement. The team given responsibility for controlling the project.

- ○ Strongly disagree (1)
- ○ Disagree (2)
- ○ Neither agree or disagree (3)
- ○ Agree (4)
- ○ Strongly agree (5)

Q21 Was the team allowed to control the project?

- ○ Not at all (1)
- ○ Input was requested but not considered (2)
- ○ Input was requested and partially considered (3)
- ○ Input was requested and given full consideration (4)
- ○ Input was requested, given full consideration and implemented (5)

Q15 Rate the following characteristics of the iteration.

	Far below average (1)	Below average (2)	Average (3)	Above average (4)	Far above average (5)
Complexity (3)	○	○	○	○	○
Code Base Familiarity (4)	○	○	○	○	○
Team's Level of Project Control (5)	○	○	○	○	○
Scale (6)	○	○	○	○	○

Q22 When addressing the tasks for this iteration or sprint, how familiar were the team members with the code base for this project?

- ○ Not at all familiar (1)
- ○ Slightly familiar (2)
- ○ Somewhat familiar (3)
- ○ Moderately familiar (4)
- ○ Extremely familiar (5)

Q29 Regarding the code base for this iteration, the team was:

- ○ Not at all familiar with the code base (1)
- ○ Slightly familiar with the code base (2)
- ○ Somewhat familiar with the code base (3)
- ○ Moderately familiar with the code base (4)
- ○ Extremely familiar with the code base (5)

Q17 How would your team rate the iteration timeline?

- ○ Not aggressive at all (1)
- ○ Slightly aggressive (2)
- ○ Moderately aggressive (3)
- ○ Very aggressive (4)
- ○ Extremely aggressive (5)

Q19 Overall, the team felt the timeline was:

- ○ Not aggressive at all (1)
- ○ Slightly aggressive (2)
- ○ Moderately aggressive (3)
- ○ Very aggressive (4)
- ○ Extremely aggressive (5)

Q31 Iteration timeline aggressiveness:

- ○ Not aggressive at all (1)
- ○ Slightly aggressive (2)
- ○ Moderately aggressive (3)
- ○ Very aggressive (4)
- ○ Extremely aggressive (5)

Q16 How would the team rate the scale of this iteration?

- ○ Very small (1)
- ○ Small (2)
- ○ Average (3)
- ○ Large (4)
- ○ Very Large (5)

Q23 What was the team's perception of the scale of the iteration?

- ○ Very small (1)
- ○ Small (2)
- ○ Average (3)
- ○ Large (4)
- ○ Very Large (5)

Q14 Rate the team's overall knowledge of the problem domain associated with this iteration.

- ○ Not at all knowledgeable (1)
- ○ Slightly knowledgeable (2)
- ○ Somewhat knowledgeable (3)
- ○ Moderately knowledgeable (4)
- ○ Extremely knowledgeable (5)

Q24 What was the team's knowledge of the business issues associated with this iteration?

- ○ Not at all knowledgeable (1)
- ○ Slightly knowledgeable (2)
- ○ Somewhat knowledgeable (3)
- ○ Moderately knowledgeable (4)
- ○ Extremely knowledgeable (5)

Q25 Regarding the business domain of this iteration, the team's was:

- ○ Not at all knowledgeable (1)
- ○ Slightly knowledgeable (2)
- ○ Somewhat knowledgeable (3)
- ○ Moderately knowledgeable (4)
- ○ Extremely knowledgeable (5)

Q26 Was technical debt considered during the effort estimation process?

As a reminder of the definition of technical debt, Steve McConnell succinctly described technical debt as a consequence of "a design or construction approach that's expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now (including increased cost over time)." In short, technical debt is a consequence of a previous development decision rather than a software bug or defect.

- O Not at all (1)
- O A little (2)
- O Somewhat (3)
- O A lot (4)
- O A great deal (5)

Q33 During any team meetings about effort estimation, did agile team members discuss technical debt?

- Not at all (1)
- A little (2)
- Somewhat (3)
- A lot (4)
- A great deal (5)

Q28 Based on your memory, was technical debt discussed during the effort estimation process?

- Not at all (1)
- A little (2)
- Somewhat (3)
- A lot (4)
- A great deal (5)

Q27 Rate the team's perception of the complexity of this iteration.

- Very complex (1)
- Moderately complex (2)
- Somewhat complex (3)
- Slightly complex (4)
- Not complex (5)

Q30 Indicate your level of agreement with the following statement:

The complexity of this iteration was high.

- ○ Strongly disagree (1)
- ○ Disagree (2)
- ○ Neither agree or disagree (3)
- ○ Agree (4)
- ○ Strongly agree (5)

Q31 Was there anything the team could have done to improve the accuracy of the effort estimate? Please explain.

_____

Q32 Thank you for participating in this study.

(Optional) If your team would like to see the final results of this study, please leave an email address where the results can be sent to you.

VITA

James Frank Ball, Jr.
dearjimmyball@gmail.com

An executive with over ten years of executive leadership experience in both corporate and higher education environments. Customer focused with the ability to balance customer requests and technology initiatives in a budget sensitive environment. Experienced in multiple phases of a project life cycle, from initial analysis to go live events and continuous client support. Effective in building and directing security and disaster recovery efforts. Skilled in leading integration implementations in the support of academic units and corporate acquisitions.

Core Competencies

- Strategic and Operational Planning
- Budget Administration
- Project Management and Execution
- Customer Service Management and Advocacy
- Policy Development
- Team Building and Recruitment
- Security and Disaster Recovery Leadership
- Technology Integration with Mergers and Acquisitions
- Software Development and Quality Assurance
- Infrastructure Implementation and Support

Higher Education Experience

Deputy Chief Information Officer for Academic Technology, University of Mississippi, January 2012 to Present

- Provides executive leadership within two divisions, namely Information Technology (IT) and Outreach and Continuing Education (Outreach)
- Leads the ongoing effort to unify the technology strategy across IT and Outreach divisions
- Administrates budget requests and prioritization of state appropriated and auxiliary funds
- Oversees the management of the Faculty Technology Development Center, Information Technology Helpdesk, Classroom Technology and Outreach Information Technology departments
- Negotiates contracts in association with the Office of General Counsel and the Office of Procurement Services
- Sets the strategy and implementation timelines for software development staff within Outreach and the Office of Research and Sponsored Programs

- Assists the Security Coordinator with advising departments on the implementation of FERPA and HIPAA standards as well as performing security audits
- Assists research faculty with resolving a variety of technology and instructional barriers while maintain appropriate security safeguards
- Assists School of Business Administration faculty with security analysis for FBI and law enforcement presentations
- Supports students using social media tools developed for early detection of technology issues
- Directs efforts for technology enhancements in the support of classroom delivery, online delivery and distance education at all regional campuses
- Develops software automation solutions to integrate systems
- Conducts annual EDUCAUSE Center for Analysis and Research (ECAR) student survey
- Participates as an EDUCAUSE proposal reviewer for the annual EDUCAUSE conference
- Serves on the Accessibility and Instructional Technology university standing committees

Adjunct Instructor, School of Business Administration, University of Mississippi, August 2015 to Present

- Provides interactive and hands-on instruction to undergraduate students, using private sector experience to expose students to career opportunities in technology
- Instructs students from multiple academic disciplines on the fundamentals of Management Information Systems
- Assists individual students to improve understand of concepts and course grades

Manager of Security and Development (Interim), The Mississippi Center for Supercomputing Research, University of Mississippi, September 1999 to February 2000

- Provided high performance computing (HPC) research support and HPC license management
- Led Year 2000 microcomputer compliance efforts for the University of Mississippi
- Managed various software development projects for the Office of Information Technology at the University of Mississippi

Supercomputer User Consultant, The Mississippi Center for Supercomputing Research, University of Mississippi, May 1993 to May 1998

- Managed information services such as World Wide Web, gopher and mailing list servers
- Provided C, C++ and Fortran programming support to scientists and software developers
- Developed web and HPC applications

Research Associate, The Jamie Whitten National Center for Physical Acoustics, University of Mississippi, May 1989 to May 1993

- Developed applications for data acquisition, scientific data analysis and data visualization
- Created end-user documentation for commercial applications

Private Sector Experience

Chief Information Officer, mTrade, LLC, April 2016 to Present

- Directs technology units including Software Development and Information Technology as well as Security implementation initiatives
- Leads the annual budget prioritization of the mTrade capital budget
- Performs vendor contract negotiations for technology initiatives
- Advises mTrade directors on automation and next generation solutions development
- Participates in the overall strategic efforts of the executive management team
- Represents mTrade concerning Information Technology services including client site performance and system availability
- Led the mTrade team concerning the technology disconnect as a part of the divestiture from FNC, Incorporated.

Chief Information Officer, FNC, Incorporated, September 2004 to January 2012

- Developed the strategies for Information Technology and Quality Control initiatives
- Directed Customer Support, XML and Forms, Quality Control, Information Technology and Data Extraction (OCR) departments
- Led personnel with annual budget prioritization of the FNC capital budget
- Advised FNC directors on next generation solutions development
- Managed technology integrations related to corporate acquisitions
- Participated in the overall strategic efforts of the executive management team
- Represented FNC concerning Information Technology services including client site performance and system availability
- Managed an in-house big data migration and application modernization effort, moving from Oracle to Microsoft platforms
- Assisted the internal enterprise risk leadership with client executive correspondence during client driven disaster recovery and security audits
- Assisted Human Resources leadership with recruiting business and engineering students

Director of Production Support, FNC, Incorporated, September 2002 to September 2004

- Advised executive management on technology initiatives and budgeting requirements for future growth to meet client expansion efforts.
- Directed various departments including XML and Forms technology, Quality Control, Information Technology (Infrastructure), Data Extraction (OCR) and Customer Support
- Led the initial software development security audit team
- Assisted the COO with the development and promotion of security and disaster recovery standards
- Developed automated solutions for system and application monitoring
- Created the first automated appraisal and broker form data extraction application within the mortgage industry

Manager of XML and Forms, FNC, Incorporated, December 2001 to September 2002

- Managed personnel and technical support efforts associated with the Appraisal Institute XML standard (AI Ready). *NOTE: AI Ready is the foundational XML standard in the mortgage industry*
- Maintained the AI Ready software and XML document standard
- Developed web-based and Microsoft desktop applications for nationwide deployment

Senior Developer, FNC, Incorporated, September 2001 to December 2001

- Developed web-based and Microsoft desktop applications for nationwide deployment
- Created software documentation
- Provided advanced technical support to FNC and customer software developers

Instructor, Batky-Howell, Incorporated (60%+ travel, contract and full-time), January 1998 to September 2001

- Trained software developers, system administrators and network analysts at corporate sites nationwide in the instructor-led and virtual class format
- Piloted, designed and delivered the initial virtual class format
- Assisted with the development of technical courses
- Consulted with corporate customers on various software tools and design strategies

Education

Doctorate of Philosophy, Business Administration
Management Information Systems Focus with Computer Science Minor Course Emphasis
2019, University of Mississippi, Oxford, MS

Master of Arts, Economics
Emphasis in Computational Economics
1997, University of Mississippi, Oxford, MS

Bachelor of Arts, Mathematics
Summa Cum Laude, 4.0 GPA in Mathematics Courses
1989, Mississippi State University, Starkville, MS

Publications and Presentations

J. F. Ball, "IT PERSPECTIVE: CAN WE AVOID THE HAMSTER WHEEL OF INNOVATION?", accepted refereed conference proceedings for the Southwest Decision Sciences Institute, Little Rock, AR, March 2017.

J. F. Ball, "Can an Evolutionary Development Teaching Model Accelerate Programmer

Competency for Beginners?", accepted refereed conference proceedings for the Southwest Decision Sciences Institute, Houston, TX, March 2015. **Recipient of a 2015 Best Student Paper award.**

J. F. Ball, "Managing Initscripts with RedHat's chkconfig", Linux Journal Magazine, April 2001, Issue #84.

J. F. Ball, "Web Security Basics with Apache: Authentication and Secure Log Files", SysAdmin Magazine, February 1999, Volume 8, Number 2.

J. F. Ball, "GIF Images on the Fly", Linux Journal Magazine, November 1997, Issue #43.

J. F. Ball, R. E. Dorsey, J. D. Johnson, "Non-linear Optimization on a Parallel Intel i860 RISC Based Architecture", Journal of Computational Economics, Volume 10, Number 3, August 1997.

K. Gates, J. F. Ball, C. Raffa, "When, Where, and What: Three Web-to-Database Applications for An Academic Environment", Presentation at 1996 CAUSE in San Francisco, CA, 1996.

K. Gates, J. F. Ball, C. Raffa, "The Lighthouse: A Lightweight Call Tracking System for Adapting to Changing Technology Needs", Presentation at 1996 ACM SIGUCCS Conference in Chicago, IL, 1996.

R. Hickling, J. F. Ball, R.K. Burrows, and M. Petrovic, "Computational Structural Acoustics, Applied to Scattering of Sound by Spherical Shells", National Center for Physical Acoustics, University of Mississippi, 1992.

R. Hickling, R. K. Burrows, and J. F. Ball, "Rotational Waves in the Elastic Response of Spherical and Cylindrical Acoustic Targets in Water", National Center for Physical Acoustics, University of Mississippi, 1990.

R. Hickling, R. K. Burrows, and J. F. Ball, "Sound Power Flow Associated with Sound Scattered by a Solid Sphere in Water", Presentation at International Congress on Intensity Techniques in Senlis, France, August 27-29, 1990.

R. Hickling, R. K. Burrows, J. F. Ball, and M. Petrovic, "Power Flow for Sound Incident on a Solid Aluminum Sphere in Water", National Center for Physical Acoustics, University of Mississippi, 1990.

R. Hickling, R. K. Burrows and J. F. Ball, "A Return to an Old Acoustics Problem, The Scattering of Sound by a Solid Elastic Sphere", Symposium in Honor of Professor T.Y. Wu at the California Institute of Technology, August 17-18, 1989.