2020

Strategies in Software Development Effort Estimation

Kevin R. Roark
Walden University

<u>Walden University</u>

College of Management and Technology

This is to certify that the doctoral study by

Kevin Roark

has been found to be complete and satisfactory in all respects,
and that any and all revisions required by
the review committee have been made.

Walden University
2020

Abstract

Strategies in Software Development Effort Estimation

by

Kevin Roark

MS, Walden University, 2018

MS, Southeastern Oklahoma State University, 2004

BS, Southeastern Oklahoma State University, 2003

Doctoral Study Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Information Technology

Walden University

August 2020

Abstract

Software development effort estimating has notoriously been the Achilles heel of the software planning process. Accurately evaluating the effort required to accomplish a software change continues to be problematic, especially in Agile software development. IT organizations and project managers depend on estimation accuracy for planning software deliveries and cost determination. The purpose of this multiple case qualitative study was to identify strategies used by software development professionals in providing accurate effort estimations to stakeholders. The planning fallacy served as the study's conceptual framework. The participants were 10 software development professionals who were actively engaged in delivering estimates of effort on software development requests in South Texas in the United States. Data were collected from 10 software development professionals in 5 different organizations. Additionally, 23 organizational documents were gathered and reviewed. Thematic analysis was used to identify codes and themes. Prominent themes were (a) defining and decomposing requirements, (b) referencing historical data, (c) identifying risks and unknowns, and (d) fostering communication, collaboration, and a consensus. A key recommendation is for software developers to ensure requirements are defined and decomposed by evaluating the request and breaking the request into manageable pieces to understand the effort required to complete the task. Implications for positive social change include improving morale, work-life balance, alignment of expectations, and software quality.

Strategies in Software Development Effort Estimation

by

Kevin Roark

MS, Walden University, 2018

MS, Southeastern Oklahoma State University, 2004

BS, Southeastern Oklahoma State University, 2003

Doctoral Study Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Information Technology

Walden University

August 2020

Dedication

I dedicate this study to my wife, Cecilia, who has been an inspiration for perseverance, and to my son Sam, whom I have tried to exemplify perseverance.

Acknowledgments

I acknowledge and offer thanks to several people for their involvement, contribution, and encouragement. Cecilia, Sam, Kathy, and Stephen supported my efforts and encouraged me to complete my research. I hope, in return, you will gain inspiration for my endeavor. Additionally, I offer my gratitude to my committee chair, Dr. Steven Case, my second committee member, Dr. Jose Feliciano, and Dr. Gary Griffith, my university research reviewer. You have challenged and encouraged me.

Table of Contents

List of Tables

Section 1: Foundation of the Study

Agile is a common approach to developing and maintaining software. The popularity of agile worldwide has increased significantly (Haines et al., 2017). However, software effort estimation is more challenging in agile projects due to changes in requirements and uncertainty experienced in the development process. Accurate estimates play a significant role in software development to identify cost and delivery schedules (Bilgaiyan et al., 2017). A software project's success relies on estimations that reflect accurate effort prediction (Bilgaiyan, Mishra, & Das, 2016). Accurate estimates increase the probability of project success; yet estimate accuracy remains an elusive target.

Background of the Problem

Software development teams have struggled to give stakeholders accurate development effort estimates, especially in the context of *agile* projects. The popularity of agile software development methodologies has increased dramatically worldwide since its inception (Haines, Idenudia, & Raisinghani, 2017). Software development professionals who use an agile development approach embrace changing requirements that often are not always fully understood in the initial development stage. Within this context of vague or changing requirements, the delivery of accurate software estimates can thus be problematic for organizations, stakeholders, and the development team (Bilgaiyan, Sagnika, Mishra, & Das, 2017). Software development groups are frequently required to give an estimate of effort at the beginning stages of product planning, before knowing the entire scope of the request or the product for which they are to create a solution (Osmanbegović, Suljić, & Agić, 2017). There are many types of estimate strategies, such as expert judgment, algorithmic models, and nonalgorithmic models (Shekhar & Kumar, 2016). In this research, I examined estimation approaches and strategies that development groups use in establishing estimates to fulfill the expectations of the product stakeholders.

Problem Statement

In software development estimation, inaccuracies of effort and time are endemic (Shekhar & Kumar, 2016) and have severe effects on software development projects (Adnan & Afzal, 2017). Accurate forecasting of effort and duration required in software development projects during their initial stages of planning increases the probability of project success (Pospieszny, Czarnacka-Chrobot, & Kobylinski, 2018). It is not uncommon for software development cost overruns to average 30% (Løhre & Jørgensen, 2016). Actual development effort can exceed initial estimations by as much as 250%, creating delays and surprises in software development projects (Dragicevic, Celar, & Turic, 2017). The general IT problem is the lack of accuracy in estimating software development effort. The specific IT problem is that some agile software development professionals lack strategies for providing accurate software development effort estimations for project managers.

Purpose Statement

The purpose of this qualitative multiple case study was to identify strategies that agile software development professionals use to provide project managers with accurate software development effort estimations. The study sample included software development professionals from five organizations who are responsible for producing effort estimates for segments of the software development process. At the time of data collection, the professionals selected for this study used an agile methodology in new and maintenance software development projects undertaken by small- to medium-sized companies in South Texas. The potential positive social impact of providing accurate software development estimates is the possible improvement of the work-life balance of those involved in software development. A more accurate effort estimation can provide project managers with the ability to project realistic delivery schedules, thus improving customer satisfaction. Accurate estimations can also potentially improve product quality, lower stress levels and improve the work lives of those involved with the software development and delivery, and provide organizations with a more realistic time expectation of software delivery.

Nature of the Study

I chose the qualitative methodology using a multiple case design for this study. The qualitative approach gives a voice to the participants, their experiences, and subjective viewpoints (Yilmaz, 2013) and therefore was appropriate for eliciting a clear understanding of successful strategies in software development estimation. The qualitative research design used herein allowed for the exploration, examination, and identification of specific estimation strategies used by development teams to deliver accurate effort estimates successfully, resulting in the delivery of software development changes within time and budgetary commitments. Researchers use a quantitative study approach to determine causation or trends and the testing of a hypothesis through statistical analysis of collected numerical data (McCusker & Gunaydin, 2015). The qualitative approach was not appropriate for the current study, as it was exploratory and not constructed to test a hypothesis using statistical methods. The mixed-methods study involves both a qualitative and quantitative approach that combines analysis of participants' experiences and statistical testing. I did not select the mixed-methods approach, as my study did not contain a quantitative component

I used a multiple case design to collect qualitative data from multiple individuals and organizations for comparison and analysis. The multiple case design can provide evidence that is more comprehensive than the information provided by a single case source (Ponelis, 2015). I sought to explore, examine, and identify strategies used by multiple software development groups. Studying a single team would not have provided sufficient data to answer the research question. I considered designs other than a case study but opted against using them. The ethnographic design is concerned with the examination, study, and understanding of a specific culture (Fayard & Van Maanen, 2015). I did not choose the ethnographic approach because the focus of my study was not on addressing the cultural issues associated with estimation. The phenomenological design provides the researcher with a means to describe participants' individual lived experiences (Ellis, 2016). Because I wanted to explore strategies rather than experiences, I opted against using the phenomenological design. Researchers use the narrative design to recount the story of the participants (McAlpine, 2016). This design was also inappropriate because this study did not involve the narration of a story. To elicit compelling evidence, data from multiple organizations were explored and examined, thus making the multiple case study an appropriate design for this research.

Research Question

What are the strategies that agile software development teams use successfully to provide their project managers with accurate estimates of software development effort?

Interview Questions

1. What are the strategies that you utilized to provide software project managers with accurate software development effort estimates?
2. What are the reasons that may cause or contribute to you or your team underestimating development effort?
3. What are the reasons that may cause or contribute to you or your team overestimating development effort?
4. Please describe the primary steps that you or your team use in the estimation process.
5. What are some factors that you consider in the software development request in providing an estimate?
6. What are some tools and techniques that you have found useful in producing accurate development effort estimates?
7. Please describe the feedback (if any) that you receive from your managers regarding your estimations?
8. What methods or processes do you find work best when proposing a development estimation?
9. What additional information would you like to share regarding effective strategies in software development effort estimation?

Conceptual Framework

In this study, I viewed the estimation accuracy of software development within the conceptual framework offered by the *planning fallacy*. The planning fallacy, identified in 1977 by Kahneman and Tversky, refers to a phenomenon where a prediction of how much time is needed to complete a task is typically optimistic. Predictors tend to underestimate the amount of time required to complete a task even if they are aware of previous estimation information (Buehler, Griffin, & Ross, 1994; Kahneman & Tversky, 1977). Estimate inaccuracies can influence software delivery timelines (Shmueli & Ronen, 2017). Concerning the planning fallacy, estimate predictions may be optimistic because people do not consider risks and setbacks (Newby-Clark, Ross, Koehler, Buehler, & Griffin, 2000). Underestimates of software development effort because of the planning fallacy can have severe consequences for the success of project outcomes (Pospieszny et al., 2018).

The purpose of this study was to identify estimation strategies used by software developers to provide accurate effort estimates in an agile development environment. Inaccurate completion times can have economic, social, and personal costs. According to the concept of the cone of uncertainty, software estimates created in the early stages of a project can underestimate actual final effort by as much as 40% (Dragicevic et al., 2017). I used the planning fallacy as a conceptual framework to understand the phenomenon whereby people typically rely on a limited number of heuristic principles to reduce the complex task of assessing probabilities and predicting values. Underestimating development effort is harmful to projects, as insufficient time and resource allocations almost invariably result in unaccomplished commitments (Khuat & Le, 2016).

Definition of Terms

Cone of uncertainty: In the initial stages of agile development, details of the final solution and defined requirements tend to be unclear. As the project variability decreases, so too does uncertainty (Krstić, Skorup, & Lapčević, 2018).

Fibonacci sequence: A series of numbers where each number is the sum of the previous two numbers—for instance, 1, 2, 3, 5, 8, 13, and 21 (Raslan, Darwish, & Hefny, 2015).

Fuzzy logic: A computational system that differs from traditional Boolean logic. Unlike the absoluteness of classical true and false, fuzzy logic provides for a degree of truth or falsehood to exist (Zadeh, 2015)

Kalman filter: An estimating and tracking algorithm that uses multiple historical data points to estimate the future value that statistically minimizes error (Soni & Kohli, 2017).

Nonfunctional requirements: Software attributes such as security, performance, and quality requests (Usman, Börstler, & Petersen, 2017).

Object-oriented: A widely accepted software development model (Rath, Acharya, & Satapathy, 2016). The object-oriented method of software development differs from a classical procedural approach (Rath et al., 2016). Objects are a combination of data and processes that correspond to real-life attributes that promotes code reuse (Saravanan et al., 2017). A *class* is the foundation of object-oriented design (Kukreja & Garg, 2017).

Stagewise development: A software development method conducted in stages: operational plan, operational specifications, coding specifications, component testing, assembly testing, and system evaluation (Misra, 2012).

Use case model: A diagram in requirement analysis that describes the behavior and sequence of action performed (Usländer, 2016).

Waterfall: A sequential, linear development approach (Stoica, Ghilic-Micu, Mircea, & Uscatu, 2016), sometimes referred to as traditional software development (Kotaiah & Khalil, 2017).

Assumptions, Limitations, and Delimitations

Exceptional circumstances and situations almost invariably affect research activity. The following assumptions, limitations, and delimitations identify conditions and events that influenced this qualitative study. According to Levitt, Motulsky, Wertz, Morrow, and Ponterotto (2016), the acknowledgment of these circumstances and situations provides integrity to a study.

Assumptions

Certain assumptions underlay this study. Assumptions are beliefs that are accepted as accurate or statements that are taken for granted (Yang, Liang, & Avgeriou, 2018). The primary assumption in this study was that the participants would answer the interview questions as accurately and truthfully as possible. The second assumption was that the participants would have an in-depth understanding of the software effort estimation process. A third assumptionm was that the use of a multiple case qualitative study approach would provide the needed data for the examination of accurate estimation strategies.

Limitations

Limitations refer to the potential weaknesses of a study. Busse, Kach, and Wagner (2017) defined limitations as imperfections of theory and methodology that do not question the validity of the findings of the study. Limitations of this study include unknown circumstances of the participants that may have biased their viewpoints and a limited number of participants interviewed. A limitation of a qualitative study is that the number of participants interviewed may not be sufficient to reach data saturation (Moser & Korstjens, 2018). This study was limited to the experiences and opinions of the participants gathered during interviews and may not be generalizable to all software development teams. Additionally, the study was limited to software development professionals in South Texas, which may limit the representability and findings of the study to the broader U.S. population.

Delimitations

Delimitations refer to the limitations set by the researcher to explicitly identify the boundaries of the research (Denscombe, 2013). The study was limited to small- to medium-sized organizations that develop software. A second delimitation was that this study only included members of a software development group who participate in the estimation process. Additionally, I restricted this study to software development groups within the area of South Texas.

Significance of the Study

Contribution to Information Technology Practice

The findings of this qualitative multiple case study may provide agile software development professionals with practical and proven strategies to deliver accurate estimates to product or project managers. Product and project managers ask developers to determine how long it will take to develop a software product. Successful strategies for a more predictive measure of software development effort may provide management with increased accuracy in delivery expectations, thus providing greater alignment with organizational budgets and customer expectations. The identified strategies may improve the accuracy of software development effort estimates in agile development teams. Realistic effort estimation processes may provide organizations with more accurate delivery expectations. The findings of this research may add knowledge to improve the IT process of software estimating and create a pathway for organizations to develop change. Improved estimation accuracy may add value to the software development practitioner, the project, and product managers, the IT organization, and stakeholders of the software product.

Implications for Social Change

If the research is successful in providing strategies that produce accurate effort estimations for software development delivery, it can potentially improve the lives of those engaged in the process of software estimation. Increased accuracy of software effort estimation could reduce the stress level of IT managers, project stakeholders, and software development professionals by providing more realistic time frames for the delivery of software. Estimates that are more reflective of actual effort can also benefit development teams by improving morale, work-life balance, alignment of expectations, and software quality. Providing development managers and development teams with practical and effective strategies for accurate effort estimations may therefore engender positive social change for a variety of stakeholders.

A Review of the Professional and Academic Literature

The literature review for this study contains analysis and synthesis of journal articles about agile software development effort estimation and related topics. I have included additional information regarding agile development methods and some common agile estimation methods. Themes addressed in the literature review are (a) agile software development, (b) current research in software development estimations, (c) the planning fallacy and optimistic bias, and (d) supporting theories and contradictory theories. The themes were chosen to provide background information on agile software development, development approaches, and estimation methods and analysis of the selected conceptual framework, the planning fallacy. In reviewing the academic literature on software development estimation, it was evident that multiple approaches, models, and strategies exist, and accurate effort estimation remains problematic.

The purpose of this qualitative multiple case study is to identify strategies that agile software development professionals use to provide project managers with accurate software development effort estimations. The goal of the literature review is to explore current strategies and methodologies in delivering software development effort estimates. The literature review contains articles gathered from ACM Digital Library, Business

Source Complete, IEEE Xplore Digital Library, ProQuest Central, SAGE Knowledge Journals, Science Direct, Taylor and Francis Online, Thoreau Multi Database Search, and Google Scholar. I used Ulrich to verify that the references in this study were peer reviewed. The study contains 263 references, of which 88% were peer reviewed and 207 were published within 5 years of my expected graduation. The literature review includes 174 articles, of which 168 (88%) were peer reviewed and 138 (79%) were published within 5 years of my expected graduation.

Agile Software Development

Agile is a common approach to developing software. The term *agile* was initially adopted in 2001 during a meeting of 17 supporters of a lightweight development process in Snowbird, Utah (Krstić et al., 2018), as a contrast to traditional plan-driven development (Abdalhamid & Mishra, 2017). The meeting resulted in the introduction of the *Agile Manifesto*, which includes 12 core values that guide the principles of agile software development (Stoica et al., 2016). The fundamental tenets of the *Agile Manifesto* ideology are that (a) individuals and interactions are valued over processes and tools, (b) working software is valued over comprehensive documentation, (c) collaboration with customers is more important than contract negotiation, and (d) responding to change rather than a defined a project plan (Coleman, 2016; Drury-Grogan, Conboy, & Acton, 2017). The agile approach to software development consists of self-organized teams with a focus on collaboration and communication (Vallon, José, Prikladnicki, & Grechenig, 2018).

Agile provides flexibility not found in typical waterfall methodologies. Agile traditionally incorporates extensive user involvement in the development process and a light touch by management (Taylor, 2016), as well as short development cycles, continuous releases, and rapidly evolving requirements (Drury-Grogan et al., 2017). Agile software development is characteristically iterative, with incremental development cycles and close communication with customers and end users (Anooja & Rajawat, 2017). The agile process has gained full acceptance among development teams in the management and construction of software.

Agile software development methodologies. The evolution of agile software management is the result of challenges with legacy development methodologies such as stagewise, waterfall, and spiral. The popularity of agile software development methodologies has increased worldwide (Haines et al., 2017). The critical processes in agility are iterative, timebound cycles that accommodate change (Boby, Kadadevaramath, & Edinbarough, 2017) and enable development teams to manage uncertainty and unforeseeable changes (Dönmez & Grote, 2018). The most widely used agile development approaches are scrum, extreme programming, feature-driven development, kanban, and the crystal family of development methods (Brad, Birloi, Bratulescu, & Blaga, 2016). Agile provides an approach to solving the issues associated with the rigidity of legacy methodologies that hindered the benefits of flexible iterations.

The agile development methodology embraces changing requirements that are often not fully understood when the project begins. Changing software requirements that are typically driven by the customer can adversely affect the quality of the final software product (Baruah, 2015). Many organizations embrace the agile methodology in response to the demand for quick delivery, reduced costs, and an increase in project flexibility (Ebert & Paasivaara, 2017; Stoica et al., 2016). The popularity of the agile software development approach has reduced time to market, increased corporate competitive advantage, and resulted in a higher level of quality satisfaction (Haines et al., 2017). The agile approach is well suited for managing uncertainty in software development (Mirzaei & Mabin, 2017), and the process offers significant benefits that include knowledge learning, employee satisfaction, confidence from feedback, and scalability (Solinski & Petersen, 2016). The agile methodology addresses the need to provide working software to customers quickly and adapt to changing requirements.

Scrum. The scrum development approach is the most popular agile method used in software development (Butt, 2016). The scrum methodology, developed by Schwaber and Sutherland (Azanha, Argoud, de Camargo, & Antoniolli, 2017; Krstić et al., 2018), was initially presented by Schwaber in 1995 at a conference in Austin, Texas (Ozierańska, Skomra, Kuchta, & Rola, 2016). Scrum utilizes incremental fixed timebound iterations in the construction of software (Ozierańska et al., 2016). The term *scrum* comes from the sport of rugby, where team members organize and collaborate to achieve the goal of winning the game (Azanha et al., 2017). The critical factors of scrum are transparency and visibility to everyone, inspection to detect problems in the early stages of development, and the ability to adjust through adaptation (Srivastava & Jain, 2017).

The scrum development method uses *sprints*, which are timebound iterations in the construction and delivery of software. Sprints are typically 2 to 4 weeks in length (Kirmani, 2017a; Torrecilla-Salinas, Sedeño, Escalona, & Mejías, 2015). At the end of the sprint, the team is expected to provide a potentially shippable working model (Mirzaei & Mabin, 2017). The development goal, as well as the development team, should not change during a sprint, and the product owner or development team may redefine the scope as needed (Srivastava & Jain, 2017). The sprint team iterations (time length of the sprint) remain constant but can vary from team to team.

Scrum has three essential elements: roles, artifacts, and events. The principal roles are the scrum master, the product owner, and the scrum development team (Hohl et al., 2018; Kotaiah & Khalil, 2017; Munawar & Qureshi, 2015). The responsibility of the scrum master is to support the scrum team, ensuring the project achieves its goals, whereas the product owner is the expert on the business case, controls the backlog, and has the power to make decisions on behalf of the company (Munawar & Qureshi, 2015). The development team is responsible for the delivery and implementation of a releasable product at the end of each sprint (Srivastava & Jain, 2017). The development team usually consists of three to nine professionals responsible for delivering a functional product and has the authority to determine the necessary actions to achieve the objectives of each sprint (Azanha et al., 2017). The team defines and sets the goals before the beginning of the sprint.

The activities (events) of scrum focus on the sprint, which is the heart of the scrum development approach. Sprints start with a sprint planning meeting that sets the goals and guidelines of the iteration (Azanha et al., 2017). Each day during the sprint, the development team conducts a stand-up meeting to report the accomplishments of the team the day before, the plan for the current day, and any impediments that the team has encountered (Abdullah & Qureshi, 2018). The benefit of the stand-up meeting is to assess the current progress and mitigate any risks that may arise (Perkusich, Gorgônio, Almeida, & Perkusich, 2017). The stand-up meeting provides the team with the ability to communicate and share project knowledge, report on progress, and resolve issues that arise during the sprint.

At the end of the sprint, the team conducts a review meeting to evaluate the accomplishments of the sprint. The sprint review meeting is a retrospective in which the team evaluates the sprint in terms of communication, resources, and processes to identify any potential areas for improvement (Srivastava & Jain, 2017). After each sprint, the team has the opportunity to share the positive and negative aspects to improve future sprints (Ahmed, Tayyab, Bhatti, Alzahrani, & Babar, 2017). The sprint retrospective is a "lessons learned" meeting to provide the team with what worked well and what did not.

The sprint backlog and the product backlog are artifacts that list the items that provide value and represent the work requested. The sprint backlog is a list of items to be accomplished in the sprint that define the requested enhancements, requirements, and corrections the team commits to working in the specific iteration (Perkusich et al., 2017). The product backlog is a priority-ordered list of everything that is needed or requested to be accomplished in future sprints (Azanha et al., 2017; Srivastava & Jain, 2017). The sprint backlog and the product backlog comprise all the requests and requirements the development team delivers to the customer.

The process of scrum starts by translating the customer's requirements into the product backlog. The team holds the sprint planning meeting with the help of the product owner to determine the planned accomplishments of the sprint (Abdullah & Qureshi, 2018). During the sprint meeting, the development team estimates the work to be accomplished (Kotaiah & Khalil, 2017; Ozierańska et al., 2016). The team transfers items planned for the sprint from the product backlog to the sprint backlog, and the team completes the items in the sprint backlog for the iteration delivery (Ahmed et al., 2017). The sprint and product backlogs are listings of items to be accomplishments as requested by the product owner.

Extreme programming. Extreme programming (XP) is an agile development method that uses on-site customer collaboration, paired programming, and automated testing processes. XP is a widely used agile method that focuses on simplicity, internal communications, and customer feedback (Singh & Pandey, 2017). XP, which was presented by Beck in 1999, is one of the oldest of the agile methods (Anwer & Aftab, 2017). According to Munawar and Qureshi (2015), the advantages of XP are short iteration cycles, direct communication with an on-site customer, and continuous integration and testing. The disadvantages are that the practice is minimal documentation and the method is not suited for projects that involve reengineering (Munawar & Qureshi, 2015). According to Anwer and Aftab (2017), the XP method is challenging to use on large projects and projects of a critical nature. XP, although practiced before the concept of agile was defined, is an agile development process.

XP, much like scrum, uses iterative phases in the software development process. The first phase of XP is the initialization phase in which project team members gather requirements from customers that are directly involved with the team to determine project scope and cost (Anwer & Aftab, 2017). In the requirements gathering phase, the team uses story cards to document and describe the request and elicit dialogue with the customer (Baruah, 2015). In the second phase (analysis phase), the software team develops the architecture and iteration plan (Anwer & Aftab, 2017). Repeated cycles of code development and testing follow the requirements and analysis phase, and the code is integrated into a deliverable release once it achieves the functional request (Anwer & Aftab, 2017). Additionally, one of the critical distinctions in the XP development process is the concept of paired programming.

Paired programming is a development approach in which two programmers sit together, one assuming the role of the driver and the other, the navigator. Paired programming is a standard practice in XP in which two people collaborate in the coding process (Hohl et al., 2018; Kotaiah & Khalil, 2017; Meyer, 2018). The driver sits at the keyboard to type the code while the navigator oversees the code input, watching for syntax errors and ensuring the program meets the required deliverable (Chen & Rea, 2018). The programmer who is actively writing the code (driver) focuses on the completion of the current task (Chen & Rea, 2018). The navigator, who is overlooking the code writing, can judge the strategic direction of the work performed, offering ideas for improvement or potential future problems (Karthiekheyan, Ahmed, & Jayalakshmi, 2018). The approach uses two programmers to collaborate on a single code set, working together to complete a development request.

Paired programming can have an advantage over developers working independently. In paired programming, the driver and the navigator often change roles throughout the project (Chen & Rea, 2018; Haines et al., 2017). The benefits of paired programming are constant code review and the brainstorming of approaches during the code's development (Karthiekheyan et al., 2018). According to Chen and Rea (2018), programmers can quickly catch and resolve errors, thus producing better code using a collaborative approach. Additionally, pairing an expert programmer with an average or novice programmer provides mentorship to the novice (Chen & Rea, 2018). However, teaming individuals who have the same expertise can cause counterproductive work (Haines et al., 2017). Therefore, the benefits of paired programming can be more significant when developers have different skill levels or experience.

Kanban. The process of *kanban,* associated with the Toyota production system, incorporates the Japanese philosophy of Muda. Muda is the avoidance or elimination of waste and the removal of activities that are not useful or do not provide value to the customer (Baseer, Reddy, & Bindu, 2015; Stoica et al., 2016). The kanban process was developed by Taiichi Ohno to provide the Toyota production system with a practical approach in specific production and market conditions and to maintain a smooth production flow to promote the concept of continuous improvement (Ahmad, Dennehy, Conboy, & Oivo, 2018). Kanban is a Japanese expression meaning *signboard* (Tanner & Dauane, 2017) and was designed as a flow control system in manufacturing in which downstream process demand signals trigger upstream process activities (Abdullah & Qureshi, 2018). The kanban philosophy, although developed for the manufacturing sector to reduce waste in product production, has been applied to software development activities.

Kanban includes a visual workflow on a board divided into columns. Teams use a kanban board to visualize the progress of work to facilitate product improvements, monitoring of processes, and effective management of the workflow (Abdullah & Qureshi, 2018; Tanner & Dauane, 2017). The purpose of the kanban board is to improve the workflow by supporting the principles of limiting work in progress, creating value throughout the process, increasing throughput, and embedding quality within the process (dos Santos, Beltrão, de Souza, & Travassos, 2018; Lei, Ganjeizadeh, Jayachandran, & Ozcan, 2017). Additionally, kanban boards provide a process to manage the workflow, balance throughput, and make processes explicit as work moves through the different states (Ahmad et al., 2018). Each state in the kanban process has a clearly defined entry and exit point and provides the team and management with a visual representation of progress.

Work requests are defined in the kanban backlog to identify the work items the team needs to accomplish. In software development, stakeholders prioritize the requests regarding importance, urgency, or value (Tanner & Dauane, 2017). Features or requests are selected and placed on the board (Abdullah & Qureshi, 2018). Each column on the kanban board limits the amount of work in progress within the column or lane (Matharu, Mishra, Singh, & Upadhyay, 2015; Tanner & Dauane, 2017). Based on prioritization, work items are pulled through the workflow using defined stages such as “to do,” “in progress,” and “done.” Work items are tasks pulled only when required (Matharu et al., 2015). Each stage limits the number of items (work in progress) to avoid the potential for bottlenecks (Abdullah & Qureshi, 2018; Dennehy & Conboy, 2017). Limiting work in progress restricts the number of ongoing activities to avoid an excess of initiated tasks and unfinished work (Matharu et al., 2015; Stoica et al., 2016). The Kanban method allows a team to respond to market changes, reduce waste, increase quality, and improve predictability.

Feature-driven development. Feature-driven development (FDD) is a development model that focuses mainly on the design and build aspects of software development. FDD is a process-oriented software development methodology used to create business critical applications and systems (Kirmani, 2017a). Luca introduced the FDD model in 1997 (Sambare, 2017) with the idea of grouping software features by categories for development (Kotaiah & Khalil, 2017). FDD is an iterative and incremental approach to software development according to functionality valued by the client and with an emphasis on quality (Nawaz, Aftab, & Anwer, 2017). The development objectives are categorized and accomplished by feature groups.

The FDD development process includes five activities that have a distinct entry and exit point. The activities are: develop an overall model, build a features list, plan by feature, design by feature, and build by feature (Nawaz et al., 2017). The FDD model focuses on the design and building processes emphasizing software quality aspects with accurate monitoring of the development project (Kirmani, 2017a). The distinction between FDD and other development methods is that the stakeholders can track project progress by a feature design, and the team builds the product using a feature aspect (Baseer et al., 2015). In FDD, the team groups software features into categorized sets for development based on functionality rather than time-bound iterations (sprints).

Planning in FDD is more extensive than that of many other agile methods. The first process in FDD is to develop an overall model that involves a discussion of the scope of the project, including the requirements of the stakeholders (Kotaiah & Khalil, 2017) and a walkthrough with the team (Nawaz et al., 2017). Following the walkthrough, the team prepares a listing of features grouped into sets that are verified against business needs and prioritized for development (Nawaz et al., 2017). The plan by feature is the third process intended to delegate the selected features to the software developers (Nawaz et al., 2017). The design by feature activity follows the plan by feature process to determine what features the team can develop within a fixed period and includes outlining the class models (Nawaz et al., 2017). The final process is the construction of the features followed by a unit test; a feature is then pushed to the main product build once the feature is complete and unit tested. (Sambare, 2017). The FDD model defines the processes from the discussion and planning to the development work and finally to adding the change to the main releasable codebase.

Adaptive software development. The adaptive software development (ASD) model, like most agile methods, assumes that change is inevitable. The ASD is a method that encourages incremental iterations using a prototyping model (Kirmani, 2017a). The ASD process model facilitates communication and planning, analysis, design and development, and testing and deployment (Sadaf, Iqbal, Saba, & Mohsin, 2017). Software development teams use the ASD method to support a component based development approach that works well with large teams and safety critical projects. Introduced by Highsmith in 2000, ASD uses a speculate, collaborate, and learn cycle rather than the traditional plan, design, and build lifecycle (Hohl et al., 2018). The ASD model is one of the earlier agile approaches.

Learning loops are a vital process in ASD. The learning cycle integrates learning loops to enhance collaboration in the goal of implementation (Hohl et al., 2018). During the speculate phase, the team gathers the requirements, and the development process begins with the schedule and the development objectives fixed (Al-Zewairi, Biltawi, Etaiwi, & Shaout, 2017). The development team works on several components concurrently, and the components are refined continuously in an iterative process (Kirmani, 2017a). However, the ASD approach does not provide for the identification of agile team members who participate in the analysis phase, the criteria for software requirements selection, or the criteria during the analysis phase (Sadaf et al., 2017). The ASD model uses timebound iterations, usually consisting of four to five-week sprints, and users participate in all iterations and face to face meetings (Kirmani, 2017a). Like most agile approaches, ASD does not put a strong emphasis on documentation.

Crystal. The crystal methodologies are a lightweight and versatile software development family of methods. Team size and project priority are the principal characterizations of crystal methods (Sambare, 2017; Tarwani & Chug, 2016). The methods were developed initially by Cockburn and are considered a lightweight development approach (Hohl et al., 2018) that promotes flexibility (Butt, 2016; Kulkarni, Padmanabham, Harshe, Baseer, & Patil, 2017). The crystal family of methods focuses on teamwork, flexibility, communication, and simplicity to improve processes (Kotaiah & Khalil, 2017).

The crystal family is identified by color to indicate the type of development. The colors include: clear, yellow, orange, and red (Kirmani, 2017a; Kulkarni et al., 2017; Sambare, 2017), indicating factors such as the size of the team, the system criticality, and the priorities of the project (Fustik, 2017). Color represents the weight or size of the project; the darker the color, the larger the project (Saravanan et al., 2017). Alqudah and Razali (2017) stated that crystal orange denotes a project with around 40 developers, whereas crystal clear is more suitable for smaller projects with fewer developers. Additionally, crystal orange is more appropriate when a high degree of rigor is necessary, whereas clear is more flexible and lightweight (Alqudah & Razali, 2017). The color indicator of the method identifies the characteristics of the project and team and counter the one-size-fits-all ideals of other software development approaches.

The crystal method family provides projects with a framework designed for development size and criticality. Crystal is one of the more adaptable methodologies (Fustik, 2017; Kotaiah & Khalil, 2017), recognizing that each project may require individual policies and processes to meet the uniqueness of the project (Fustik, 2017; Sambare, 2017). The principles of crystal are passive knowledge transfer, continuous delivery, frequent releases, and automated testing (Hohl et al., 2018). The crystal family of methods arose from the need for an approach that was customizable to accommodate differences in projects.

Dynamic system development method. The dynamic system development method (DSDM) grew out of the need to provide a standard for the rapid application development process (Kirmani, 2017a; Tarwani & Chug, 2016) before the term agile was coined (Brad et al., 2016). The DSDM method, introduced in 1994 (Kirmani, 2017a; Sadaf et al., 2017), and according to Tarwani and Chug (2016), credit Van Bennekum with conceiving the development methodology. Like most agile methods, DSDM focuses on business value, active user involvement, frequent delivery, integration testing, and collaboration with stakeholders (Fustik, 2017). However, DSDM, unlike many agile methods, provides complete support throughout all life cycle phases (Kirmani, 2017a). The DSDM philosophy is that the team can deploy 80% of the system in 20% of the time (Kirmani, 2017a) with the possibility of rework and that development changes must be reversible (Fustik, 2017).

Requirement priority determines the most critical functionalities to deliver first in DSDM. The requirements are prioritized and checked for feasibility (Baruah, 2015). Project requirements are prioritized based on the rules of must-have, should have if possible, could have but not critical, and will not deliver now but maybe later (Younas, Ghani, Jawawi, & Khan, 2016). DSDM has three phases: the pre-project, the project life cycle, and the post-project phase. The pre-project phase established the goals and priorities of the project (Fustik, 2017). In the project life cycle phases, the functional model, design, iteration, and implementation phases are determined (Fustik, 2017). The post-project phase addresses functional efficiency and error correction (Fustik, 2017). The DSDM approach sets the time allotment and resources and adjusts the amount of functionality delivered accordingly (Kirmani, 2017a).

Estimation of Effort in Software Development

Estimation in software development is the process of approximating how much effort is required to accomplish a task. Estimation plays a significant role in software development to establish cost assessments and delivery schedules (Bilgaiyan et al., 2017). A software project's success relies on estimations that reflect accurate effort prediction (Bilgaiyan, Mishra, & Das, 2016). According to Rahikkala, Leppänen, Ruohonen, and Holvitie (2015), software projects that delivered expected results within budget and predicted time are the exception rather than the rule. Accurate effort estimations provide stakeholders with forecasting data for planning, budgeting, and project scheduling. Estimating effort in a software project continues to be problematic for development teams.

Multiple factors contribute to inaccurate effort estimations. Factors that adversely affect software development effort estimation are the uncertainty of the effort required to complete the task, software size, estimator experience, inconsistent and incomplete data, the dependency of the environment, and frequent changes in requirements (Sehra, Brar, & Kaur, 2016). Tanveer (2017) concludes that estimation accuracy is dependent on the developer's experience, complexity, and the impact of changes made to the underlying system. Additionally, estimation models perform differently in different environments and development project types (Sehra, Brar, Kaur, & Sehra, 2017). Considering multiple factors, estimation of effort in software development can thus be difficult, and underestimation is problematic.

Multiple factors affect software development estimation. Influences that affect estimation accuracy are software size, the team’s experience, the team's skill, the number of nonfunctional requirements, the distribution of the team geographically, and the level of communication provided by the customer (Usman et al., 2017). Sehra et al. (2016) asserted that factors that affect accurate estimation are: requirements uncertainty, software size, the experience of the estimator, incomplete data, and changes in requirements. Jørgensen (2014) stated that the accuracy of estimates improves through the use of local context, the use of historical estimation error intervals, the avoidance of misleading estimation information, the use of a checklist, the conducting of a group-based approach, and avoidance of early estimation based on incomplete information. Multiple factors, both individually and collectively, adversely effect estimation.

Providing accurate effort estimates in software development is problematic. Estimating effort in software development projects at the beginning of the lifecycle is more challenging due to the “cone of uncertainty” (Sehra et al., 2016). In the initial feasibility stage, actual effort and cost can exceed 250% more than the initial estimate (Dragicevic et al., 2017). Delaying the estimation until the requirement specification phase can reduce inaccuracy, thus providing a more realistic and accurate estimation (Sehra et al., 2016). However, estimations many times are requested before the elaboration of requirements. As a project progresses, uncertainty decreases as knowledge increases regarding the product (Arifin, Daengdej, & Khanh, 2017). Estimations are predictions, and there is a level of uncertainty as each project is unique, and there are no two projects with the same requirements.

Software effort estimation is more challenging in agile projects due to changes in requirements and uncertainty experienced in the development process. Rahikkala et al. (2015) identified two factors associated with estimates that positively influenced project success. The first factor was that senior management ensures that a software estimate relies on facts rather than guessing or opinion (Rahikkala et al., 2015). The second was that senior management recognizes that estimates are critical to organizational success (Rahikkala et al., 2015). Accurate estimates increase the probability of project success; yet estimate accuracy remains an elusive target.

Agile estimation methods. There are two primary categories of software estimation methodologies. All estimation approaches are either algorithmic (parametric) or non-algorithmic (nonparametric) or a combination of the two (Idri, Amazal, & Abran, 2015; Osmanbegović et al., 2017; Soni & Kohli, 2017). Algorithmic approaches utilize mathematical models or equations, whereas non-algorithmic do not (Khuat & Le, 2016). Estimating provides planners with project timelines and costs.

There are multiple approaches used by development teams to provide estimates of effort. Shekhar and Kumar (2016) asserted that no single method in software development estimation is considered the best method, and they suggested using a combination of techniques to increase estimation accuracy. Shekhar and Kumar (2016) concluded that it is best to use non-algorithmic approaches such as an expert judgment for projects that have extensive known requirements. For projects with many unknowns, the algorithmic approach is the more appropriate choice. However, estimations that use a combination of methods arrive at a more accurate estimate (Shekhar & Kumar, 2016).

Cost estimation is essentially forecasting the expected time, effort, and workforce needed to complete the development of a software task or project (Bilgaiyan et al., 2017). Popular estimation methods used are estimation by analogy, expert judgment, function points, software sizing, and Bayesian methods (Bilgaiyan et al., 2016; Soni & Kohli, 2017). Jørgensen (2014) asserts that estimates should use simple models, historical data, and should avoid misleading information. Additionally, Jørgensen (2014) stated that estimates could be improved by utilizing checklists, utilizing structured approaches, and avoiding early estimations.

The philosophy of agile effort estimation is that the people doing the work perform the estimation to gain a more realistic assessment (Taylor, 2016). Prakash and Viswanathan (2017), Bilgaiyan et al. (2017), and Osman and Musa (2016) concurred that different estimation models are better suited to different development models. Distinct characteristics of successful agile estimating include collaboration with product owners, estimations accomplished by a team rather than an individual, and the use of story points for relative measures (Prakash & Viswanathan, 2017). In the early stages of development, obtaining refined details of the project may not be possible in an agile environment, making estimation problematic.

Story points. Story points are a sizing technique used as a relative unit of measure for expressing the overall size of a user story or development effort. The utilization of story points is the most popular estimation approach for software sizing to measure the effort needed to implement a user story (Choetkiertikul et al., 2018; Dragicevic et al., 2017). The estimation method is a bottom-up approach and provides a measure of the complexity or quantity of work to produce (Jadhav, Shaga, & Thorat, 2017).

There is no fixed formula for defining the effort or size in the utilization of story points (Osman & Musa, 2016). Story points commonly utilize the Fibonacci sequence to express relative size (Alostad, Abdullah, & Aali, 2017; Jadhav et al., 2017; Raslan et al., 2015). The gaps between the sequences provide a higher degree of uncertainty in the level of effort for larger units of work. Essentially, the larger the effort (greater the size), the more likely the error in the estimate; thus, the higher the gap in the sequence (Raslan et al., 2015). Fox (2016) claimed that the Fibonacci sequence, when used as an estimation metric, is relatively unbiased. Usman et al. (2017) suggested an extended approach to story points by providing estimates by averaging three values; fastest, most practical, and maximum values to give a final estimate.

Story points are relative values, can differ from team to team, and are numerical representations of complexity. The estimating approach (size value) is specific to each team and uses each team's cumulative knowledge (Choetkiertikul et al., 2018). The story point value can change from team to team depending on the baseline story in which they are relative too (Choetkiertikul et al., 2018; Soni & Kohli, 2017). Each development team uses story points on a different scale to establish a velocity over time (Ahmed et al., 2017). Story points are relative measures rather than quantitative measures (Soni & Kohli, 2017). The accuracy of story points is subjective to the person or persons performing the estimation and derived from previous experience (Arifin et al., 2017).

Values are determined from previous efforts using a relative approach and differ from team to team.

One of the critical success factors for story point estimation is that the development group estimates the stories as a team using the same scale. Once the story points are estimated, the values are translated into the team's velocity to forecast future sprints, iterations, or efforts (Brad et al., 2016). Velocity represents the total of story points that the development team can deliver in a specific iteration (Torrecilla-Salinas et al., 2015), and is a useful predictor of the team's capabilities (Ahmed et al., 2017). Story points are independent of time units and are a successful and common approach in software development estimation (Zahraoui & Idrissi, 2015). Harzl (2017) indicated that there could be disadvantages to using story points as a critical factor in the success of story point estimation is a team's shared experience. According to Harzl (2017), it is challenging to establish velocity in the initial iterations, as team members have not had experience working and estimating as a collective group and experience challenges in providing accurate estimates. The story pointing approach, although commonly used, is subject to error.

T-shirt sizing. T-shirt sizing is an estimation approach that utilizes relative valuations. The estimation approach uses t-shirt sizes such as extra-small, small, medium, large, and extra-large (Alostad et al., 2017; Raslan et al., 2015). Similar to story points, the t-shirt sizing approach can differ from team to team and requires a common understanding of the estimated value selected (Alostad et al., 2017). The strategy works best when a team has estimated previous stories or work items as a group, and the method can provide a measurement for large effort work items (Alostad et al., 2017; Raslan et al., 2015). The t-shirt sizing technique can produce an early estimate to give the business a metric of complexity for determining the level of effort (McConnell, 2006). The early comparison allows stakeholders or requesters to determine if the effort required is worth the business value generated from the effort (McConnell, 2006). The t-shirt size estimation technique offers a simple alternative to executing a more complex estimation process.

The advantages of t-shirt sizing are that as there are fewer values to select and the voting process can be conducted expeditiously. According to Harzl (2017), due to the abstract nature of the approach, t-shirt sizing does not suggest precision. However, with t-shirt sizing, sizes are non-numerical, and the approach is simple and easily understood (Harzl, 2017). The t-shirt sizing method provides a nontechnical, initial estimation projection that is accurate enough to support effective project control (McConnell, 2006). The disadvantages in using the approach are that velocity is hard to measure (performance of the team over time), the scale lacks detail, there is no precise mathematical correlation between the sizes, and a numerical value to track effort actuals is lacking (Harzl, 2017). Thus, the absence of numerical values is problematic in establishing velocity.

Expert judgment. Expert judgment in software effort estimation requires someone with previous experience in effort estimation who knows and understands the task under consideration to provide an approximation of effort. Expert judgment utilizes the knowledge of an expert and is a widely used strategy for software estimating (Shekhar &

Kumar, 2016). However, expert judgment can exhibit bias by the estimator and relies on the expert's previous experience on similar projects to generate a realistic estimate (Khuat & Le, 2016). Expert judgment comprises two approaches: effort-time and effort-size (Arifin et al., 2017). Effort-time is an absolute value method, such as person-days or person-hours; effort-size is a relative measure such as story points (Arifin et al., 2017) or t-shirt sizing. McConnell (2006) states that using a top-down approach that decomposes tasks into a granularity that is less than about two days enhances the accuracy and effectiveness of expert judgment. Large task estimation is prone to error and more challenging to estimate; thus, decomposition provides higher accuracy.

Expert judgment is ta common estimation technique used in effort estimation in software development. Although there is high availability of commercial estimation tools and approaches, expert estimation remains the most widely used estimation methodology (Ivan & Despa, 2016; Shekhar & Kumar, 2016; Usman, Britto, Damm, & Börstler, 2018). Expert-based effort estimates result from quantitative intuition as experts seldom base estimates on explicit analytical argumentation (Jørgensen & Boehm, 2009). Expert judgment is a non-algorithmic technique and may be prone to error as estimations can be inconsistent, lack repeatability, and be overly dependent on human memory (Sehra et al., 2017). Estimation inaccuracies can stem from over-optimism and over-reliance on accuracy due to over-confidence in the estimator's ability to deliver accurate estimations.

Delphi. The Delphi technique utilizes a consensus-based approach to estimating involving multiple experts. The experts selected for a Delphi approach have subject domain experience and specific application knowledge (Adnan & Afzal, 2017; Strasser, 2017). Experts conduct discussions in a structured group process designed to produce a consensus (Osman & Musa, 2016; Perkusich et al., 2017). The Delphi method of estimation in software development is the process whereby a group of experts identify the task to estimate, provide an estimation method, discuss the application of the method, and arrive at a consensus regarding the level of effort needed (Rai, Gupta, & Kumar, 2017; Strasser, 2017). The experts conduct the approach using multiple rounds of voting that provides results that can be evaluated and summarized (Lima, West, Winston, & Wood, 2016). The experts may repeat the process of estimate revision until the experts reach a specific number of rounds, reach a consensus, or until the results are stable and the answer is satisfactory (Nguyen, & Nguyen, 2018; Prakash & Viswanathan, 2017). The Delphi approach to estimation makes available an estimate based on the collective agreement of experts.

The Delphi approach utilizes expert assessments and involves the coordination of the team and elaboration of requirements for the members of the team to do their estimations anonymously. Estimations with a high level of variation are discussed further and re-evaluated (Prakash & Viswanathan, 2017; Strasser, 2017). The results are distributed to the group for further discussion after each round to reach an agreement and review the agreement for relevance (Bilgaiyan et al., 2016). The Delphi method captures factors from several experts and provides a defined practice in the assessment (Lee & Rothenberger, 2015). The Delphi approach is the collective assessment of experts to establish an agreed-upon estimation.

COCOMO II. The COCOMO II model is an algorithmic approach to estimating software development effort. COCOMO II uses size and numerical input measures regarding application points multiplied by constants that are empirically determined to provide estimations (Ivan & Despa, 2016). The use of company-specific calibration and historical data increase accuracy (Moharreri, Sapre, Ramanathan, & Ramnath, 2016). The COCOMO II model has the advantages of objectivity, repeatability, built-in sensitivity to development factors, and model calibration to previous projects and experiences (Osmanbegović et al., 2017). COCOMO II uses multiple factors for calibration and is most effective when using historical data.

The algorithmic COCOMO II estimation model's effectiveness relies on historical data to provide accurate estimations. Estimators calibrate the model using factors such as flexibility of the development, team cohesion, reuse, architecture, risk, platform experience, database size, the volatility of the platform, personnel continuity and experience, time constraints, complexity, and team capability (Boehm et al., 2000). An advantage of COCOMO II is that modification and customization of the model are straightforward (Prakash & Viswanathan, 2017). However, Prakash and Viswanathan (2017) also stated that the method becomes much less effective if historical data is not available. Additionally, the COCOMO II model is more suited to a procedural development paradigm than the agile development model (Kukreja & Garg, 2017; Rath et al., 2016).

Bayesian network. Bayesian networks (BN) belong to the category of probabilistic graph models and are used to represent knowledge about uncertain domains (Perkusich et al., 2017). Bayesian networks represent a joint probability distribution over a set of variables (Freire, Perkusich, Saraiva, Almeida, & Perkusich, 2018). Dragicevic et al. (2017) suggested that the BN model is a suitable estimation method in an agile software development methodology as it does not have an impact on agility and can be applied in an early planning phase successfully. The BN model is useful in making predictions and diagnostics with ambiguous data to determine the probability of an event (Dragicevic et al., 2017). Estimators use the method to incorporate causal factors to determine conditional probability is estimations.

The BN is a model that describes probabilistic relationships between causally related variables. The advantages of a BN are suitability for small projects, and it provides results based on incomplete data sets (Zare F., Zare H., & Fallahnezhad, 2016). The BN model's additional advantages are the explicit treatment of uncertainty and support for decision analysis (Perkusich et al., 2017). The use of BN can be advantageous in effort estimation because probability distributions can be updated as new information becomes available, and estimation models are constructed using causal influences (Perkusich et al., 2017). Bayesian networks allow for the combining of historical data with expert opinion.

Planning poker. Planning poker is a widely used estimation method for agile software development teams (Prakash & Viswanathan, 2017; Soni & Kohli, 2017; Usman et al., 2017). The estimation method uses a consensus approach to estimate development effort that minimizes peer pressure (Taylor, 2016) and is useful if historical data is not available (Anooja & Rajawat, 2017). The first step in planning poker is a domain expert explaining the user story to the team and providing clarification if requested (Lopez-

Martinez, Ramirez-Noriega, Juarez-Ramirez, Licea, & Martinez-Ramirez, 2017; Torrecilla-Salinas et al., 2015). The next step is the creation by the team's members of a private preliminary estimate followed by the display of their estimations to the entire team, typically using cards that represent a value (Torrecilla-Salinas et al., 2015). Team members explain the reasoning for estimations, and each member reflects on the other explanations (Miranda, 2017). Additional estimation rounds may be needed if estimates differ significantly (Miranda, 2017; Torrecilla-Salinas et al., 2015). The estimators that provide the highest and lowest values explain their reasoning, and the team continues with subsequent rounds until it reaches a consensus, and an agreed upon amount is determined (Torrecilla-Salinas et al., 2015; Vyas, Bohra, Lamba, & Vyas, 2018). Planning poker can consist of several rounds of discussion and re-estimation to reach consensus (Bilgaiyan et al., 2017; Choetkiertikul et al., 2018). Much like the Delphi method, developers use a collective forum in the planning poker technique, and open discussions provide a group-based agreement to the estimate.

The estimation methodology is a team-based exercise used for assigning a relative estimate value to a requirement that expresses the level of effort required to deliver the specific feature. Planning poker traditionally uses the numerical sequence such as the Fibonacci sequence (Ramirez-Noriega, Juarez-Ramirez, Navarro, & Lopez-Martinez, 2016). Planning poker is a standard estimation approach and requires expert opinion and analogy (Osman & Musa, 2016; Usman et al., 2017). Planning poker estimations are consensus-based and result in a value or size estimation of effort.

The planning poker method is most effective when an expert is engaged in the estimation and when the team has previous experience with similar tasks. Planning poker was introduced by Grenning (Rai et al., 2017) in 2002; the technique combines expert opinion, analogy, and disaggregation into a quick and reliable estimation method. The goal of planning poker is to arrive at an estimation that will withstand future scrutiny (Osman & Musa, 2016). Planning poker is an incremental team-based method that collectively analyzes requirements and determines an estimation (Dönmez & Grote, 2018). The distinct difference between planning poker and Delphi is that not all group members in a planning poker session are required to be experts.

Artificial neural networks. Artificial neural networks (ANN) are used as data analysis tools and are known for their learning and generalization ability (Shawky, Salwa, & El-Hafiz, 2016). ANN is a mathematical model (algorithmic) inspired by biology (Mittas, Papatheocharous, Angelis, & Andreou, 2015). Neural networks provide relationships between complex data through a learning phase (Rijwani & Jain, 2016). Types of neural networks used are general regression networks, polynomial neural networks, and probabilistic neural networks (Prakash & Viswanathan, 2017). ANN uses processing features called neurons, each having a mathematical function with specific inputs, a computational procedure, and outputs (Rijwani & Jain, 2016). According to Kaushik, Tayal, Yadav, and Kaur (2016), ANN models used in software estimation are the radial basis function network (RBFN) and function link artificial neural network (FLANN). The RBFN model offers a straightforward design, good generalizability, strong tolerance to noise, and learning ability (Kaushik et al., 2016). The FLANN method is suited when data is nonlinear and is less complicated (Kaushik et al., 2016). Although ANN is considered an algorithmic process, the network itself is not an algorithm, but rather a framework of learning algorithms.

ANN's principal characteristic is the ability to approximate nonlinear functions and is thus similar to traditional statistical techniques such as logical regression, statistical regression, and discriminant analysis (Mittas et al., 2015). The ANN method utilizes machine learning and pattern recognition for estimation and can discover relationships between the dependent and independent variables (Kaur, 2017). Artificial neural networks have gained popularity for software estimation prediction due to their ability to capture complex data and to disregard noise in the input data (Pospieszny et al., 2018). ANN uses data from previous software projects to provide outputs by inference through learned data (Rijwani & Jain, 2016). The ANN design, inspired by the biological nervous system processes information using computational elements (nodes) operating through weighted inputs (layers) to provide accurate estimates (Bilgaiyan et al., 2017; Mittas et al., 2015). Additionally, the more considerable the amount of historical data, the more accurate the estimation; thus, the ANN is most effective in achieving accurate software development estimations when historical data is available (Naik & Nayak, 2017).

Function points. The function point (FP) method calculates an estimate using the parameters of inputs, outputs, inquiries, and files. The technique was introduced by Albrecht in 1979 (Hans & Gahlot, 2016) to measure the size of data processing systems from the end user's point of view to determine an estimated development effort

(Abualkishik et al., 2017). FP's advantage is that estimators can calculate effort when a defined use case or in-depth system analysis is not available (Dewi, & Subriadi, 2017a). Estimators calculate function points by summing the number of internal logical files, external interface files, external inputs, external inquiries, and external outputs (Hans & Gahlot, 2016; Yoshigami, Tsunoda, Yamada, & Kusumoto, 2017). Function points are numerical values that measure software size determined from data processing types rather than from software development complexity.

Prakash and Viswanathan (2017) stated that the FP method is appropriate to estimate size and cost but cannot estimate effort. Function points represent the amount of functionality released to the user by determining the data transactions, and operations that involve data crossing the boundaries of the application (Abualkishik et al., 2017). The FP method provides an estimation method that allows managers to project software size early in the project life cycle (Qi et al., 2017). It is independent of the technology used in the development of the software project (Farah-Stapleton, Auguston, & Giammarco, 2016).

COSMIC. To overcome some of the early issues with function point measurements, a group of experienced software measurement experts formed the Common Software Measurement International Consortium (COSMIC). The COSMIC standard defines rules and principles for measuring software's functional size (Almakadmeh, Al-Sarayreh, & Meridji, 2018). The COSMIC method differs from the traditional function point method, as the focus is on data movements such as input, output, and data storage that characterize most software development efforts (Di Martino, Ferrucci, Gravino, & Sarro, 2016).

The COSMIC method is a second-generation function point method proposed to overcome a few shortcomings of the function point method. The COSMIC process counts data movements classified into data entry and exit points, which are input-output movements, and read and write of data to storage (Almakadmeh et al., 2018). Each data movement is one COSMIC function point of size in a software application. COSMIC function points are the sum of the sizes of the functional processes (Abualkishik & Lavazza, 2018). The higher the number of data movements, the more significant is the size of the software.

Use case points. Use case points (UCP) is a technique that utilizes a UML use case diagram to estimate the size. The UPC technique, inspired by the function points method, is an appropriate method to use in the early stages of software development (Azzeh & Nassif, 2016). Karner developed the technique in 1993 as an estimation method for object-oriented software (Shollig, Widodo, Sutanto, & Subriadi, 2016). Mehta and Kumari (2016) suggested that a technique such as the UCP estimation is more appropriate in object-oriented development than function point counting and COCOMO. The method calculates complexity based on use cases (Shollig et al., 2016) and thus differs from the calculation of data movements (function point) and historical data (COCOMO).

Azzeh and Nassif (2016) state that the first step in using the UCP technique is to calculate the unadjusted actor's weight or complexity of interaction, such as simple, average, or complex. The second step is to classify the transaction using the same scheme of simple, average, or complex (Azzeh & Nassif, 2016). Transactions are a response between an actor and the system. Finally, the transaction and complexity values are adjusted based on technical complexity and an environmental adjustment factor (Azzeh & Nassif, 2016). The basic UCP calculation is UCP = (UUCW + UAW) * TCF * ECT where UUCW is unadjusted use case weight, UAW is unadjusted actor weight, and TFC and ECF are technical complexity and environment factors respectively (Urbanek, Kolcavova, & Kuncar, 2017).

The UCP model has been used broadly in recent decades (Rath et al., 2016), and studies have indicated the method's reliability (Dewi, & Subriadi, 2017b). However, according to Azzeh and Nassif (2016), the major disadvantage is that values are arbitrary in calculating software size, making it challenging to provide time-based estimations. Time-based effort and size are not directly proportional to each other (Rath et al., 2016). The UPC concept utilizes documented use cases in the determination of size.

Conceptual Framework – The Planning Fallacy

People making predictions tend to underestimate the time it will take to complete a task. Kahneman and Tversky (1977) identified the concept of the "planning fallacy," a phenomenon where a prediction regarding how much time will be needed to complete a future task is usually optimistic. Kahneman and Tversky (1977) indicated that overconfidence increased with ignorance (Kahneman & Tversky, 1977). People's insensitivity to evidence quality (reliability of information available) and predictions based on small sample sizes contribute to overconfidence (Kahneman & Tversky, 1977).

Kahneman and Tversky (1977) further stated that the contributing factors to overconfidence were the assumption of normal conditions and anchoring.

Previous research by Kahneman and Tversky (1973) stated that in making predictions, people do not appear to follow statistical results, but instead, they make predictions based on intuition. Additionally, in 1974, Tversky and Kahneman (1974) describe cognitive bias that stemmed from judgmental heuristics such as representativeness, availability of scenarios, and anchoring. People typically rely on a limited number of heuristic principles, which reduces the complex task of assessing probabilities and predicting estimation (Kahneman & Tversky, 1973). Kahneman and Tversky (1973) proposed that people have an insensitivity to the prior probability of outcomes, sample size, and a misconception of chance. People also have an insensitivity to random events that may affect the estimation and a perceived illusion of validity in providing estimates (Kahneman & Tversky, 1973). Thus, people provide estimates based on the assumption and do not consider possible events that may cause a potential delay.

Two types of information are available when predicting tasks' duration: singular and distributional (Kahneman & Tversky, 1977). Distributional is primarily a consideration of previous task performance, whereas singular focuses on the task itself (Kahneman & Tversky, 1977; Thomas & König, 2018). The planning fallacy is the result of underestimation as a consequence of neglecting or ignoring distributional data resulting in an error in prediction (Kahneman & Tversky, 1977). Neglect of distributional data could be the result of the perceived uniqueness of a project (Kahneman & Tversky, 1977). Research conducted by Zhu, Li, Yang, and Xie (2019) concluded that an increased focus on additional information (distribution) led to later predictions, but a focus on task content (singular) resulted in earlier predictions.

Kahneman and Tversky (1977) identified three conclusions in describing the planning fallacy:

1. Errors in judgment are many times more systematic than they are random.
2. The presence of bias is frequent in both experts and non-experts.
3. Judgments should be driven from a reflective assessment rather than from immediate impressions, although intuition from a knowledgeable professional is beneficial (Kahneman & Tversky, 1977).

Additionally, Tversky and Kahneman (1974) demonstrated that it is common to adjust an estimate due to an anchoring effect. The establishment of an anchor can adversely influence an estimation prediction. Predictions are often based on an optimistic view of the duration of a previous task and are not adequately adjusted for the demands of a new task that is to be estimated (Tversky & Kahneman, 1974). The anchoring effect is the influence of initial information that influences the estimator's judgment, including information that may be irrelevant (Løhre & Jørgensen, 2016). Anchoring effects estimation as it has an influence on a judgment from an initial presented value.

Buehler et al. (1994) explore the phenomena of the planning fallacy and explain why people underestimate task completion times. The evidence suggests that individuals believe that their project will proceed as planned even while knowing that a clear majority of projects run late (Buehler et al., 1994). People base predictions on a plan for carrying out a task and formulate their predictions on the assumption of positive events occurring rather than adverse events (Wiese, Buehler, & Griffin, 2016). The absence of consideration of unforeseen events adversely affects the accuracy of estimates.

Conclusions reached by Buehler et al. (1994) suggest that people make more realistic predictions when they use past experiences to inform their predictions (distributional). However, people also focus on the details of the specific case (singular) rather than distributed information about a related set of cases (Buehler et al., 1994; Tversky & Kahneman, 1974; Wiese et al., 2016). They tend to hold to a belief that their project will proceed as planned (singular) even knowing that a clear majority of previous projects (distributional) have faltered and run late (Thomas & König, 2018). Kahneman and Tversky (1973) identified three heuristics in making estimation judgments under uncertain conditions: judgments based on representativeness, availability of scenarios, and estimation adjustments based on anchors. In generating an estimate, people often have a perceived illusion of the time required to complete a task and a false sense of validity to their estimation.

Over-optimism, resulting in underestimation, is an identified problem in the prediction of effort. Buehler, Peetz, and Griffin (2010) asserted that people are typically optimistic in their estimates and predict that they will finish projects earlier rather than later. Buehler et al. (2010) test their hypostasis on both closed and open-ended tasks to determine if predicted task completion times influenced actual completion times. The results indicate that making optimistic predictions may lead to finishing the task sooner (Buehler et al., 2010). Although over-optimism may result in expediting the completion of tasks, Buehler, Griffin, and MacDonald (1997) found that in both a laboratory and field environment, that incentives to complete a task early increase the effects of the planning fallacy.

The planning fallacy is the problematic phenomenon of time underestimation. Kahneman and Tversky (1973) described three judgmental heuristics: representativeness, availability, and anchoring, leading to bias in judgments. A firm reliance on judgmental heuristics precipitates inaccurate estimates in software development (Løhre & Jørgensen, 2016). Shmueli, Pliskin, and Fink (2016) found that software developers tend to underestimate project effort in the time required for project completion, resulting in one of the most common reasons for project failure.

According to Shepperd, Waters, Weinstein, and Klein (2015), people tend to display excessive optimism in their predictions that is often quite unrealistically positive. In their research, Shepperd et al. (2015) identified two types of unrealistic optimism. The first type is unrealistic absolute optimism that refers to an unjustified belief that a more favorable outcome will occur even when quantitative data indicates otherwise (Shepperd et al., 2015). The second type of optimism is unrealistic comparative optimism, which refers to one's outcome being more favorable than that of a peer (Shepperd et al., 2015).

Accurate estimations in the planning phase of software development improve the likelihood of project success. Shmueli and Ronen (2017) noted that both software developers and managers are subject to the planning fallacy resulting in the tendency to plan additional work. Time underestimation and benefit overestimation occur during planning iterations due to the planning fallacy (Shmueli & Ronen, 2017). The planning fallacy phenomenon occurs when the individual is focusing on the inside view of a task (singular) but not considering the data from an outside view of previous tasks (distributional) (Thomas & König, 2018). Additionally, even those aware of statistical regression have an inclined bias towards these heuristics in making judgments in a context of uncertainty (Kahneman & Tversky, 1973).

The planning fallacy is the problematic phenomenon of time underestimation. Shmueli et al. (2016) found that software developers tend to underestimate project effort in the time required for project completion, resulting in one of the most common reasons for project failure. Time underestimation and benefit overestimation occur during planning iterations as a result of the planning fallacy (Shmueli & Ronen, 2017). The planning fallacy phenomenon occurs when the individual estimating the effort considers only the inside view of a task (singular) but does not consider the outside view (distributional) (Thomas & König, 2018). The distributional is essentially a previous task performance, whereas the singular focuses on the task itself.

The findings of a study conducted by Shmueli et al. (2016) provide evidence of manifestations of the planning fallacy in software development projects. Shmueli et al. (2016) provide evidence of the planning fallacy in software development projects by identifying effort and time underestimation, scope overload, and over-requirements. They argue that scope overload and over-requirements are results of underestimation (Shmueli et al., 2016). The conclusions of the study suggest that although reference class forecasting and using a consultant positively influence scope overload and over-requirements, there was little to no effect on underestimation (Shmueli et al., 2016). The planning fallacy, a behavioral economic theory, advances the understanding of poor planning in software development projects (Shmueli et al., 2016; Shmueli & Ronen, 2017).

Shmueli et al. (2016) describe two views for determining the future cost of software development: the outside and inside view. The outside and inside view correspond to the singular and distribution views described by Kahneman and Tversky (1977). The outside view is the consideration of past projects' experience and knowledge to reference similar cases (Shmueli et al., 2016). The inside view is the examination of information specific to the project or task and the uniqueness of the case at hand (Shmueli et al., 2016). An inside view is a bottom-up approach that discounts historical data, past experiences, and environmental factors that potentially affect the project (Pinto, 2013). Although developers tend to estimate effort based on an inside view, the outside view provides a more accurate estimate (Shmueli et al., 2016).

The inclusion of historical effort estimation information in future estimations give the potential for greater accuracy in software development estimating (Shmueli et al., 2016). Jørgensen (2014) stated that the accuracy of estimates improves through the use of local context, historical information use, and the avoidance of early estimation based on incomplete information. When prompted to consider a task from an outside observer's perspective, people are more willing to consider obstacles that they may not otherwise have considered (Wiese et al., 2016). Additionally, the motivation for aggressive schedules and optimism of a high performing team can lead to underestimating the time needed to complete a project (Prater, Kirytopoulos, & Ma, 2017). Aggressive schedules and the neglect of an outside (distributional) view lead to inaccurate estimates.

Shmueli et al. (2016) examined the outside view approach in reducing behaviors associated with the planning fallacy in software development effort estimation and concluded that the inclusion of an outside view minimizes the problem of time underestimations, scope overload, and the over-requirements of software development. Utilizing the descriptive behavioral theory, Shmueli et al. (2016) concluded that knowledge of cognitive bias resulting from planning fallacy could mitigate estimation errors in the planning of software development projects. The results of the study showed that problems associated with time underestimation, scope overload, and over-requirements are reduced but not eliminated by presenting reference information regarding past completion times (Shmueli et al., 2016). Additionally, outside consultants can reduce the planning fallacy effects by using an outside view (Shmueli et al., 2016; Shmueli & Ronen, 2017).

Although many researchers have studied underestimation and effort over-optimism, realistic effort estimation remains problematic (Jørgensen, 2016), as software developers are usually over-optimistic and underestimate the needed effort to accomplish a task (Dragicevic et al., 2017). Software development effort underestimations may result in cost overruns and cause customers to cancel projects, and project teams may be required to work without financial compensation (Kirmani, 2017b). Additionally, the quality of the product cannot be guaranteed (Qi et al., 2017). The effects of over-optimism resulting from the planning fallacy phenomena are detrimental to the planning and estimating of software development effort.

Many software project estimations fall short of actual effort. Overwhelming evidence indicates that there is a tendency to underestimate software effort, on average of about 30% (Jørgensen, 2014). It is difficult to predict the size of a software project during the initial phases (Shida & Tsuda, 2017) due to incomplete or inaccurate requirements (Dragicevic et al., 2017). Resulting changes to requirements has a cascading effect on the software's cost and delivery time (Bilgaiyan et al., 2017). During the initial phases of software development, it is difficult to predict the project's size resulting from inaccurate, incomplete, and dynamic requirements due to changes that occur during the development cycle. In consideration of the planning fallacy phenomena, incomplete or inaccurate requirements affect the reliability of distributional (outside view) data, thus making estimation potentially unreliable and accuracy problematic.

The phenomenical effects of the planning fallacy are evident in software effort estimation. Researchers have identified contributing causes of estimation inaccuracies such as optimistic bias and the lack of or neglect of distributional information. A longitudinal case study conducted by Usman et al. (2018) concluded with the following observations about software effort estimation. First, underestimation is common and that teams with less experience produce higher estimation overruns (Usman et al., 2018). Usman et al. (2018) also stated that single-stage estimation approaches reduce accuracy, and the colocation of development group improves estimation accuracy. There are four primary causes of estimation inaccuracies in software development. Reasons are (a) optimistic assumptions, (b) unanticipated requirements, (c) a corporate culture that confuses targets with estimates, and (d) arbitrarily deadlines. Uncertainty exists in software development estimation because of human differences, market forecasting, and value judgments (Arifin et al., 2017). Anooja and Rajawat (2017) suggested that factors such as improved estimation training and higher accuracy of information (requirements) provide positive effects on effort estimation. Conventional wisdom indicates that estimates improve as projects progress (Arifin et al., 2017). As projects progress, additional information (distributional) data become available, thus improving the estimation process.

Team size effects estimation. Staats, Milkman, and Fox (2012) state that underestimation increases as a team size increases. The larger the team, the more likely the team will underestimate the tasks associated with a project primarily due to a rise in the loss of productivity due to extra process controls (Staats et al., 2012). Staats et al. (2012) state that the coordination complexity, diminished motivation of the team, and increased conflict within the team negatively affect productivity. Additionally, the increased overhead of team coordination negatively adds to underestimation.

Mitigating the planning fallacy. According to Kahneman and Tversky (1977), there are five steps involved in mitigating the planning fallacy.

1. The selection of a reference to identify a known outcome (Kahneman & Tversky, 1977)
2. The assessment of the distribution of the reference class such as the range or average (Kahneman & Tversky, 1977)
3. An intuitive estimation that distinguishes from other cases based on an expert's singular information (Kahneman & Tversky, 1977)

4. An assessment of predictability or consideration of the potential accuracy of the estimation (Kahneman & Tversky, 1977)
5. Correction for non-regressiveness in the event the intuitive estimate differs considerably, or predictability is judged as low (Kahneman & Tversky, 1977)

Additionally, Buehler, Griffin, Lam, and Deslauriers (2012) demonstrated that third-person imagery has positive effects on reducing underestimation. The finding suggests that when people consider an estimation from a third person's perspective, optimistic bias is less likely due to the use of an underlying psychological process that invokes a neutral observer (Buehler et al., 2012).

The planning fallacy and optimistic bias are observed phenomena in software effort estimation. Jørgensen (2004) states six estimation principles to reduce human estimation bias:

1. Evaluate estimation accuracy, an increased perception of accuracy can lead to decreased estimation accuracy (Jørgensen, 2004).
2. Avoid conflicting estimation goals, such as estimation for a bid or estimates based on best-case scenarios (Jørgensen, 2004).
3. Request justification form estimators, estimators are typically not skilled in the discovery of estimation weaknesses (Jørgensen, 2004).
4. Avoid information that is irrelevant or unreliable; utilize checklists.
5. Use data from previous projects, apply analytics rather than memory, use distributional information (outside view) (Jørgensen, 2004).

6. Use estimators with expert domain background and a proven track record of accurate estimations (Jørgensen, 2004).

Estimators can improve estimations by attending training on estimating. Shepperd, Mair, and Jørgensen (2018) conducted a study concluding that estimations provided by software development professionals that participated in a workshop reduced judgment bias. The study found that there are strong effects of anchoring in software effort estimations, de-biasing workshops are beneficial and reduce the variability in estimates (Shepperd et al., 2018). Moreover, the knowledge of bias and the understanding of strategies in reducing bias can improve the accuracy of estimates.

Reviewing the estimations of other software developers has a positive effect on estimation. Jørgensen (2004) stated that reviewing other software developer's estimates triggered reflection (distributional) on how much effort similar tasks required. Additionally, Jørgensen (2004) indicated that developers tend to rely on an inside view and their memory rather than background information such as distributional completion times for similar tasks. Estimation models that use historical data remove the potential bias from those that do not consider previous estimates on similar tasks.

People make more realistic predictions when they reflect on previous experiences to inform their predictions. The outside view or reflection in prior experiences is usually more accurate as it bypasses political and cognitive bias (Fridgeirsson, 2016). People also focus on the details of the specific case rather than distributed information about a related set of cases (Buehler et al., 2010). An inside view leads to a narrow focus, thus disregarding additional information such as past experiences of similar tasks (Zhu et al., 2019), and most individuals and organizations tend to focus on the inside view (Flyvbjerg, 2006). The propensity to focus on an inside view results in the planning fallacy.

Andersen, Samset, and Welde (2016) offered suggestions to improve estimations, including (a) transparency, (b) careful examination of estimations based on uncertainty analysis, (c) increased provisions for scope changes and unspecified contingencies, (d) the utilization of reference projects in creating estimates, (e) third-party review of estimates, and (f) attention to estimates formulated on incentives. Wiese et al. (2016) conducted a study on backward planning to counter optimistic bias. Wiese et al. (2016) described backward planning as starting a plan at the end and working through the required steps in reverse chronological order. Breaking large tasks into smaller subtasks highlights critical steps that are potentially overlooked otherwise (Wiese et al., 2016). The study conducted by Wiese et al. (2016) concluded that identifying obstacles is more apparent when using the backward planning approach and results in less optimistic predictions.

Reference class forecasting is the outside view based on knowledge of the actual performance of referenced comparable projects. Flyvbjerg (2006) introduced the concept of reference class forecasting to improve the inaccuracy resulting from bias by considering the actual performance of similar projects, thereby bypassing the effects of optimistic bias and strategic misrepresentation. Flyvbjerg (2006) described three steps in reference class forecasting.

1. The identification of similar projects that are broad enough to be statistically meaningful and narrow enough to be comparable (Flyvbjerg, 2006)
2. The establishment of a probability distribution within the reference class to add statistical meaning (Flyvbjerg, 2006).
3. The estimator uses a comparison of a project with a reference class distribution to establish a more likely outcome for the specific project. The outside view provides a mechanism to bypass cognitive bias (Flyvbjerg, 2006).

Reference class forecasting attempts to bypass human bias by relying on historical data from similar past projects as a guideline for predicative estimations. An accepted mitigation strategy for optimistic bias is Flyvberg's reference class forecasting that was developed and based on Kahneman and Tversky's outside view (Prater et al., 2017). Reference class forecasting is the systematic method for using an outside view when creating forecasts of similar projects rather than focusing only on the project at hand (Fridgeirsson, 2016). Reference class forecasting has a positive effect on accurate estimations as it considers an outside distributional view.

Empirical testing supports the effectiveness of reference class forecasting in reducing time and cost overruns in large projects (Wiese et al., 2016). Reference class forecasting improves effort estimation accuracy in the initial stages of planning (Fridgeirsson, 2016). Shmueli et al. (2016) found that software effort estimators can mitigate the effects of the planning fallacy by using reference information about historical completion times and by having the estimator adopt the roles of a consultant, both of which are outside views. The consideration of historical data is more likely to bypass a cognitive bias in decision making (Féris, Zwikael, & Gregor, 2017). However, learning from previous estimation mistakes does not reduce prediction bias when the current task differs from the previous task, although task similarity reduces bias (Thomas & König, 2018). Accurate estimations using reference class forecasting requires task similarity.

Flyvbjerg et al. (2018) claim that cost overrun (downstream effect) is a consequence of underestimation (upstream cause). Flyvbjerg et al. (2018) state that (a) utilization of reference class forecasting, (b) de-biasing estimations, (c) creating incentives that encourage teams to stay on budget, and (d) using a team with a proven track record of delivering within budget mitigate cost overrun. Sting, Loch, and Stempfhuber (2015) reported in their study of engineers and noted that presenting a visual cue (red card) when the engineer was having trouble reduced the potential time overage of a task. Although the red card approach does not mitigate the planning fallacy, it minimizes the phenomenon's effect (Sting et al., 2015). Engineers that request help when encountering an unknown, or experienced a risk that was unaccounted for, mitigate a potential delay in time.

Additional Theories in Effort Estimation

Anchors. The anchoring effect is the misprediction of tasks' durations due to false memories regarding previous, similar tasks. As a result of the anchors' influence, subsequent judgments can be biased even when presented with a value that may not be relevant to the judgment in question (Løhre & Jørgensen, 2016). Anchoring can create artificial scheduling heuristics, as it acts as a stake in the ground and becomes the basis from which initial estimates and subsequent modifications originate (Pinto, 2013). Anchoring is the perceived duration of a previous task that becomes a basis for establishing a prediction of a future task in which the future task prediction has not been appropriately adjusted based on differences in the future task (Thomas & König, 2018). Even professional expertise is not sufficient to avoid the anchoring effect, as the memory of the anchor comes to mind and often becomes automatically considered despite the source (Tomczak & Traczyk, 2017). In quantitative estimation, the anchoring effect is a phenomenon where an initial arbitrary number can affect subsequent numerical estimates. Although anchoring is typically related to numerical quantifiers, nonnumerical anchors have an adverse effect as well (Jørgensen, 2016).

Lorko, Servátka, and Zhang (2019) evaluated the effects of anchoring on estimations and provide evidence that numerical anchors influence duration estimates and that anchors continue to persist if estimators do not receive feedback. Results of the study suggest that when estimators are isolated from potential biasing information, they review historical estimation information, and by making the estimators aware of estimation mistakes, the effects of anchoring are reduced (Lorko et al., 2019). Additionally, Shepperd et al. (2018) concluded that the anchoring had a significant adverse effect on software development estimation. However, providing training to estimators on the impact of bias suggest a reduction in the anchoring effect (Shepperd et al., 2018). Thomas and König (2018) propose that estimators can reduce anchoring when they consider performance on previous tasks and have experience completing previous similar tasks.

Subsequent tasks are affected by the anchoring of an initial task. A bias established in the first task can serve as an anchor for the tasks that follow (Roy, Burns, & Radzevick, 2019). Even when using anchored values that are not reasonable, the anchoring effect still exists (Tomczak & Traczyk, 2017). Additionally, Løhre and Jørgensen (2016) stated that anchors negatively affect the accuracy of software development estimation, even if the anchors are implausible or unrealistic. However, they also noted that software developers with more experience are affected less by anchoring (Løhre & Jørgensen, 2016).

Optimism bias. Optimism bias is the tendency to underestimate or ignore the probability that an adverse event will occur. Kahneman and Tversky (1977) identified optimism bias as a behavioral characteristic of underestimation. Prater et al. (2017) identified optimism bias as a significant cause of unrealistic project schedule development. Optimism bias is the belief that there are fewer project risks and an assumption of a more favorable outcome, even in the face of historical information that is contradictory (Pinto, 2013). Optimistic bias can result in underestimation of task effort as unforeseen events are not considered or acknowledged.

Prater et al. (2017) state that optimism is, by its nature, a positive human trait that sets us apart from other species. Additionally, Prater et al. (2017) indicate that most research on optimistic bias concludes that reference class forecasting and the outside view are the most effective strategies for mitigating optimistic bias. People are prone to optimism and perceive that their future as more positive than another person (Polonioli, 2016). Evidence suggests that task complexity increases; underestimation becomes more apparent (Lévy-Garboua, Askari, & Gazel, 2018). Additionally, people learn to be overconfident faster than they learn their actual ability (Lévy-Garboua et al., 2018).

People tend to be more optimistic than pessimistic. Lovallo and Kahneman (2003) state that the inclination for over-optimism stems from an exaggerated perception of our talents, a misunderstanding of the degree of control we possess, the downplaying of the possibility of uncontrolled events, and the understatement of the probability of risk. Additionally, Lovallo and Kahneman (2003) identify anchoring and organizational pressure promoting a sense of optimism. Mitigation strategies include the utilization of reference class information and forecasting using an outside view (Lovallo & Kahneman, 2003). Francis-Smythe and Robertson (1999) state that there is evidence of a correlation between time management skills and an accurate estimation of effort. People who perceive themselves as good managers of time provide more accurate estimates than those who do not see themselves as good managers of time.

Optimistic bias is more prevalent in the estimation of one's effort. Many studies on human judgment prove that people are generally over-optimistic in predicting their performance (Jørgensen, 2004). Buehler et al. (1994) concluded that people have a propensity to underestimate their effort but not the effort of others. People tend to focus on plan-based scenarios rather than on past experiences (Buehler et al., 1994). They are likely to dismiss past poor performance under the belief that others caused previous problems and, therefore, do not warrant serious consideration (Buehler et al., 1994). Additionally, Yamini and Marathe (2018) claim that optimism bias can harm employee job satisfaction and increase job-related stress because of unrealistic and prolonged completion times.

Optimistic bias is a result of estimators having an overoptimistic view of essential project parameters. Wiese et al. (2016) conducted a study on backward planning as a strategy to counter optimistic bias. Backward planning involves starting a plan at the end and working through the required steps in reverse chronological order (Wiese et al., 2016). Wiese et al. (2016) suggested breaking larger tasks into smaller subtasks to highlight critical steps that are otherwise potentially overlooked. The study concluded that people more readily identified obstacles when the backward planning approach is utilized and results in less optimistic predictions (Wiese et al., 2016). In an overview of agile software development methods, according to Osman and Musa (2016), combining estimation techniques may reduce optimism in the estimation of software effort.

Overestimation of one's abilities has a direct effect on early phase estimates. According to Andersen et al. (2016), initial estimates by the person requesting the project are prone to bias. Usman et al. (2018) conducted a longitudinal case study concluding that the underestimation as a result of optimistic bias is typical in both the software analysis phase and the quotation phase. Compounding the establishment of effort estimations, the development of software is not always straightforward. Thus, the bias in software development estimation can occur and cannot be prevented entirely (van Vliet & Tang, 2016). Optimistic bias is the phenomenon of focusing on the best-case scenario and not considering potential risks, unforeseen events, or setbacks.

The hiding hand. The hiding hand is the phenomenon in which a person takes on a project with little to no knowledge or consideration of future obstacles. The theory proposed by Hirschman (1967) claims that once a project is underway and encounters obstacles, the creative action occurs, and positive results emerge. The hiding hand suggests that estimators tend to be overly optimistic, and poor planning can make decision-makers believe that projected costs are lower than the actual cost (Hirschman, 1967). However, underestimations increase creativity to overcome obstacles that are not planned or foreseen and to think out of the box, and resultingly, positive results are accidentally achieved (Ika, 2018). The principle of the hiding hand can benefit projects as over-optimism can promote creativity.

Essentially, Hirschman (1967) stated that the hiding hand is the underestimation of both costs and benefits in project appraisals. Unexpected circumstances create acts of innovative problem-solving (Anheier, 2016). The hiding hand proposes that planners tend to be overly optimistic and believe themselves to be at less risk of experiencing negative consequences than are others (Ika & Söderlund, 2016). Hirschman (1967) proposed that the hiding hand is beneficial, as it stimulates creativity and problem-solving. Human ingenuity overcomes difficulties. It can often provide unexpected benefits by justifying projects that may otherwise not be undertaken had the early difficulties been better understood (Lepenies, 2018; Room, 2018). The hidden hand's principle concept is that optimism caused by ignorance of difficulty can lead to projects that otherwise might not have been started had the real challenges been known.

Contradictory Theories

Malevolent hiding hand. Flyvbjerg (2016) disputed Hirschman’s (1967) concept of the hiding hand. Flyvbjerg (2016) argued that rather than a benevolent hiding hand, a malevolent hiding hand is more typical and pervasive. The malevolent hiding hand, as described by Flyvbjerg (2016, 2018), proposes that creativity does not overcome cost overruns and difficulties and that benefit overruns are much less prevalent than cost overruns. According to Flyvbjerg and Sunstein (2015), the driving forces of the malevolent hiding hand are ignorance, psychology, and power: ignorance of the knowledge of the problem faced; psychology regarding initial optimism; and deliberate underestimations to improve chances of project approval and funding (Flyvbjerg & Sunstein, 2015). The hiding hand is a phenomenon of unexpected circumstances invoking innovative problem solving (Hirschman, 1967). In contrast, the malevolent hiding hand is the knowledge of the potential of unforeseen circumstances yet disregards or hides the consequences (Anheier, 2016). Flyvbjerg (2016, 2018) stated that the hiding hand is less common than Hirschman theorized, and that optimism bias, cost underestimation, and benefit overestimation are more prevalent. Jørgensen (2014) indicated that underestimation is evident in competitive price markest as lower estimates are more likely to win contracts providing further evidence intentional estimation inaccuracies.

Strategic misrepresentation principle. Flyvbjerg (2013) states that estimates in the initial stages of a project are the most critical in determining whether the project will proceed or not, and be successful. However, many times, forecasts of cost and benefit are highly inaccurate (Flyvbjerg, 2013). Flyvbjerg (2013) and Parent (2019) indicate that there are two causes of estimation inaccuracies: optimism bias resulting from the planning fallacy and strategic misrepresentation. The strategic misrepresentation principle is an intentional underestimation of effort rather than unintentional optimistic bias. Flyvbjerg (2013) states that strategic misrepresentation is the deliberate misstatement of project planners' estimations by providing project stakeholders with estimates that are known to be incorrect. A study conducted by Naess, Andersen, Nicolaisen, and Strand (2015) interview respondents indicated that strategic misrepresentation is widespread and results from economic and political reasons. Additionally, multiple researchers have identified strategic misrepresentation as problematic in IT projects (Parent, 2019; Shmueli et al., 2016). According to Parent (2019), strategic misrepresentation in information technology projects stems from the fear that if project approvers knew the actual costs upfront, they would never approve the plan.

The principle of strategic misrepresentation refers to the intentional incorrect calculation of facts in favor of political or personal interests. Flyvbjerg (2018) stated that there are often political motivations in the underestimation of projects regarding the strategic misrepresentation principle. Underestimation can be motivated to ensure funding for projects from top management (Pinto, 2013). This strategic underestimation is also the result of psychological, political, and economic factors (Andersen et al., 2016). Misrepresentation can occur when forecasters provide information that intentionally overestimates the benefits or underestimates the effort of a project (Fridgeirsson, 2016). The presence of strategic misrepresentation rather than optimistic bias is more common in projects where political pressure is high (Flyvbjerg, 2006). Planners and promoters underestimate costs and overestimate benefits to increase the likelihood that the project will gain financial backing and approval.

Parkinson's Law. Deadlines can influence individual performance. Parkinson's law is the phenomenon that work expands to fill the time that is available for its completion (Brodsky & Amabile, 2018; Jørgensen, 2014; Kim & Nembhard, 2018). The observed phenomenon of Parkinson's law is that work rates increase as the remaining time shortens and that deadlines are known to increase productivity as the availability of time decreases (Izmailov, Korneva, & Kozhemiakin, 2016; Kim & Nembhard, 2018). Thus, when deadlines are further away, work speed is slower than work speed as the proximity to the deadline becomes closer (Kim & Nembhard, 2018; Kim J. E., Nembhard, & Kim J. H., 2016). Considering the phenomenon of Parkinson's law, tasks are less likely to finish early and, more likely, to finish on time.

In support of the concept of Parkinson's law, two pitfalls may exist to completing a task early. The excess time may be used to gold plate or improve the product beyond what is requested or necessary (Izmailov et al., 2016). The second pitfall is that an overestimation may seem by the administration as excessive (Izmailov et al., 2016). Thus, workers would have no incentive to ensure that potential future overestimations are untouched or reevaluated (Izmailov et al., 2016). Brodsky and Amabile (2018) provide evidence indicating that the work pace increases when tasks have deadlines providing evidence of Parkinson's law phenomenon. In the absence of deadlines or time pressure, people tend to work slower (Brodsky & Amabile, 2018).

The effects of Parkinson's law may result in tasks taking longer than expected. According to Zhang, Jia, and Diaz (2018), Parkinson's law and the phenomena identified as the student syndrome contribute to project delays and increased project costs. Jørgensen (2014) proposes that estimations may be harmful to a software development schedule and that postponing or eliminating estimates reduces the effect of Parkinson’s' law. Additionally, high estimates result is a loss of productivity of the development team (Jørgensen, 2014). Although overestimation or time buffers may offset schedule overruns, the result can be detrimental to optimized performance.

Student syndrome. Project or task overruns can result from a phenomenon called the student syndrome. The student syndrome suggests that in the beginning phase of a task, urgency is less and gradually increases as the scheduled completion time gets near (Izmailov et al., 2016). Mirzaei and Mabin (2017) observed that there were three adverse effects of the student syndrome phenomena. First, due dates and milestones often needed to be extended (Mirzaei & Mabin, 2017). Second, as the due date or milestones approached, there is a surge in activity to complete it (Mirzaei & Mabin, 2017). Finally, once the person or team completes the activity or reach the milestone, there is a downturn in productive activity (Mirzaei & Mabin, 2017). According to Zhang et al. (2018), the possibility of early completion of tasks due to the wasting of disposable time allocated for the task’s completion is lost. The effects of the student syndrome, much like the effects of Parkinson's law, result in a task completed on time at best and often are delivered late.

Groupthink. Estimations that are group-based can develop groupthink, which can have a negative result in estimates (Drury-Grogan et al., 2017). Groupthink occurs when group members strive for unanimity over their personal opinions, thus, altering the decision trajectory (Kakar, 2018; Riccobono, Bruccoleri, & Größler, 2016). Estimations provided by group discussion likely focus on the success of completed tasks based primarily on optimism (Buehler, Messervey, & Griffin, 2005). Additionally, inaccurate estimates can result from the misconception of group consensus resulting from not considering the views of all team members (Drury-Grogan et al., 2017).

Groupthink also results in the phenomena known as the Abilene paradox. The Abilene paradox refers to the problem in which each group member incorrectly believes that others in the group have a specific opinion, leading the group to a public agreement and private disagreement (Browne, Appan, Safi, & Mellarkod, 2018). Cunha, Moura, and Vasconcellos (2016) identified the Abilene paradox in software development groups and described the phenomena where groups make decisions that are contrary to the beliefs or desires of the individual members. The negative results of groupthink occur when the members override their personal opinion in favor of unanimity (Riccobono et al., 2016). The team consensus is not the result of choice, but rather the result of an implied decision by the team.

Estimating travel time. Although people tend to underestimate the time required to accomplish a task, when it comes to estimating the time it takes to travel to a destination, they tend to overestimate. Tenenboim and Shiftan (2018) state that for travel times, people focus on a subset of times that include variability resulting from previous delays. Regarding the time it takes to travel, people generally remember longer times (Tenenboim & Shiftan, 2018). The study indicates that overestimation travel times were two and a half times more prevalent that underestimating travel times (Tenenboim & Shiftan, 2018). Although underestimation is a more typical human trait (Prater et al., 2017), in the estimation of travel times, overestimation is more the norm and is contrary to the planning fallacy phenomena.

Transition and Summary

Section 1 presents an introduction to the problem of accuracy in developing software development effort estimates. In the literature review, I have discussed some of the more common software development approaches and estimation methodologies. Additionally, I have addressed the planning fallacy, a phenomenon describing over-optimism in task-time estimation. The purpose of this study is to explore strategies to improve effort estimations in software development. For this study, I have chosen a qualitative approach to answer the research questions of identifying effective estimation strategies. The planning fallacy provides the conceptual framework for this study to describing optimism bias and potential causes. The literature review discusses the issues of estimation inaccuracy in software development. This study explores strategies to reduce estimation error and provides the software development community with practical strategies to mitigate error inaccuracy.

Section 2 describes the procedures and methods used in this study and justifies the selection of the research method. The next section identifies the researcher's role, a description of the criteria for participant selection, and a justification of the choice of the multiple case design. Additionally, section 2 discusses ethical research, the approach used to analyze the results, reliability, and validity of the findings. Section 3 describes the results of the study and conclusions drawn from the qualitative analysis of the collected data.

Section 2: The Project

In Section 2, I present the reasons for selecting a multiple case qualitative study approach for this research. I restate the purpose, explain my role in the research process, and describe the population from which the sample was drawn. I justify the criteria for the selection of the sample population and the ethical considerations for the research. Finally, I describe the data collection technique, the organization and analysis of the data, and its reliability and validity.

Purpose Statement

The purpose of this qualitative multiple case study was to identify strategies that agile software development professionals use to provide project managers with accurate software development effort estimations. The study sample included software development professionals from five organizations who are responsible for producing effort estimates for segments of the software development process. At the time of data collection, the professionals selected for this study used an agile methodology in new and maintenance software development projects undertaken by small- to medium-sized companies in South Texas. The potential positive social impact of providing accurate software development estimates is the possible improvement of the work-life balance of those involved in software development. A more accurate effort estimation can provide project managers with the ability to project realistic delivery schedules, thus improving customer satisfaction. Accurate estimates can also potentially enhance the quality of the product, lower stress levels and improve the work lives of those involved with the software development and delivery, and provide organizations with a more realistic time expectation of software delivery.

Role of the Researcher

As the researcher, my role was to recruit the participants for the study; conduct interviews and collect data; and examine, analyze, and present the findings. According to Yilmaz (2013), the role of the researcher in a qualitative study is to be objective in portraying the data, providing impartiality to the study and maintaining an outsider's point of view. The researcher's role also includes gathering data and developing an understanding of the phenomenon in the study (Starcher, Dzubinski, & Sanchez, 2018). Blalock (2018) asserted that the role of the researcher is a crucial part of qualitative research as it shapes the design and analysis of the study. Seixas, Smith, and Mitton (2018) state that the role of the researcher is to describe the reality of the participant and solicit an informative description of their experiences. I was personally involved as the interviewer for the study and conducted all of the participant interviews. In conducting research, it is essential, especially in data collection through interviews, to recognize the potential for bias and take appropriate steps to mitigate any prejudice (Yilmaz, 2013). My goal was to develop interview questions that would provide insight and reflect the issues of the research problem and to engage with the participants in such a manner as to acquire the information without affecting the results.

I chose a semistructured interview as the method of acquiring data about the strategies used by the participants in estimating effort in software development. Brown and Danaher (2019) defined the semistructured interview as a data collection method whereby the interviewer has a prepared topic and list of questions to ask but which provides the latitude to elicit open-ended responses from the participants to allow a conversation to develop that may not be anticipated. The semistructured interview process provides the researcher with a method to obtain the participants' perspective and their experience regarding the research topic (McIntosh & Morse, 2015). Drury-Grogan et al. (2017) indicated that researchers commonly record semistructured interviews, follow up on insights derived during the interview, and transcribe the recording for analysis. I conducted and recorded the interviews and gathered the data while consciously trying to avoid the introduction of bias, personal beliefs, and any preconceptions about the study. Once I completed the interview process, I transcribed the recordings, looked for common trends and patterns in the data, and followed up on any information that may have required clarification.

I recognize that my previous experience as a software development manager and project manager has the potential to inject bias into the study. I have worked with software development teams for over 15 years, and I selected the area because of my familiarity with the domain. Being aware of personal opinions and predispositions will help prevent bias in a study (Cypress, 2017; Fusch, Fusch, & Ness, 2018). Recognizing that previous personal experience could influence the interview process, I attempted to structure the research questions such that the questions would not lead the participants or influence their responses. Additionally, I did not have any prior relationships with the participants or with the organizations in which the participants worked. To reduce bias in this study, I did not make the participants aware of my previous experience as a software development manager.

Participation in the study was voluntary, and I protected the identity of the study participants and their organizations. I conducted this study in an ethical manner using the National Institutes of Health's guidelines and principles for ethical research. The Institute's guidelines include respect for subjects, establishment of human subject protections, the safeguarding of participants' privacy and confidentiality, and provisions for full disclosure (National Institutes of Health, n.d.). Additionally, I adhered to the principle tenets of the *Belmont Report*: respect for persons, beneficence, and justice (The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research, 1979). I conducted this study as an independent observer, gathering data through interviews and documents. I recorded and transcribed verbatim the interviews conducted. Additionally, I documented any behavioral actions, participant reactions, and unconscious body responses of the participants during the interviews.

The purpose of the interviews was to gain knowledge by exploring the experiences and perspectives of the participants. A well-developed interview protocol is an essential element in getting useful data (Ismail, Ismail, & Hamzah, 2018). An interview protocol provides the researcher with guidance to remain focused during the interview (Arsel, 2017). Arsel (2017) observed that an interview protocol offers control to the process and a framework for translating the research questions into a natural conversation. Additionally, an interview protocol minimizes digression during the interview (Arsel, 2017). According to Fusch et al. (2018), an interview protocol can reduce bias and mitigate the interviewer's personal opinion regarding the research topic in the data collection process. I used an interview protocol as a framework to maintain uniformity in my interview process, mitigate bias, and ensure the interview questions followed a consistent pattern. Use of an interview protocol allowed me to establish a consistent course of action and uniform procedures throughout the data collection process.

Participants

In a qualitative study, the researcher must determine the criteria for participant selection to meet the objectives of the study. Participant selection and identification are essential criteria in providing breadth, depth, and saliency for authentic analysis to give validity to the study (Saunders & Townsend, 2016). Windsong (2018) stated that qualitative researchers do not use random samples as there is a logical selection of participants and location based on a specific strategy. Qualitative interviewing requires careful selection of participants and ensuring the participants know about the topic to ensure rigor in the study (Starcher et al., 2018). Qualitative research participant selection involves seeking out participants who have explicit knowledge and experience of the phenomena under examination (Flannery, 2016). The participants in this study were software development professionals from small- to medium-sized companies who engage in providing effort estimation. The participants had at least five years' experience working in a software development team either as a developer, manager, or project lead. The selected participants had familiarity with techniques for estimating effort and providing estimates considered by their project or program manager as accurate.

Inclusion in this study required confirmation of all participants' knowledge, experience, and utilization of strategies in accurately estimating development efforts.

I identified companies engaged in agile software development in South Texas through searches in my LinkedIn network. Ponelis (2015) and Stokes, Vandyk, Squires, Jacob, and Gifford (2019) identify that the use of personal networks as a valuable source for subject matter experts and research participants. Peticca-Harris, DeGama, and Elias (2016) indicate that a useful method to gain access to potential individuals for study participation is by contacting employees or managers of companies to assist in providing potential study candidates. I contacted the identified companies and requested permission to interview team members that are active in software development. Once I identified potential participants, I prescreened the candidates to affirm that they had estimation knowledge and have successfully used strategies in accurately estimating software development efforts. The participants answered "yes" to all the prescreen questions (see Appendix A) for inclusion in the study. I sent out invitation e-mails (see Appendix B) to candidates selected from the prescreening process to inform them of the purpose of the study. I contacted each selected participant by telephone. I confirmed their knowledge and experience in the estimation process and had strategies for accurately estimating effort for software development requests. Additionally, the participants selected have indicated that estimation accuracy is an essential element in software development planning, forecasting, and cost estimating. The participants affirmed that the project or program manager considers their estimation strategies as accurate.

Establishing rapport with the participants is crucial during the interview to create trust and to enable the respondent provide a rich and detailed response to the interview questions (McGrath, Palmgren, & Liljedahl, 2018). Arsel (2017) advises starting the interview dialog with warm-up questions such as “How has your day been” and “Tell me about yourself,” as well as sharing with the participants your personal story regarding the project to build rapport. Brown and Danaher (2019 suggest that the researcher establish an open dialog in the interview to develop a rapport with the participants and gain trust by showing a genuine interest in the participants’ opinions. Additionally, using responsive listening techniques such as verbal and non-verbal cues to express assent build rapport (Brown & Danaher, 2019). I conducted the interviews face to face and through online meetings with the participants to establish a rapport to gain their trust. Once I established rapport with the participants, I began with a brief discussion of the study and confirmed the participants' consent. I allowed the participants to ask questions regarding the intent of the research and provided them with an opportunity to resolve any uncertainties about the interview questions or process. I discussed with each participant the confidentiality and protection of any identifying data that I may obtain to ensure their privacy and anonymity during and after the interview.

The interview location for the participant was free of distraction and interruptions. Flannery (2016) indicated that the interview setting should be the natural setting in which the phenomena occur, and suggested that a relaxed environment will allow participants to feel at ease during the interview. Before conducting the interviews, I arranged to meet each participant to provide them with the study’s background. Each participant was allowed to suggest a location or online communication application that they would find most comfortable. I conducted the interviews at a time indicated by the participants to not disrupt the participant's work schedule.

Research Method and Design

This study's focus was to investigate strategies for estimating effort accurately in software development using a qualitative approach. In the initial stage of a study, the researcher should select the most appropriate method to adequately answer the research question (Venkatesh, Brown, & Sullivan, 2016). A qualitative method can offer an understanding of organizational behavior (Jonsen, Fendt, & Point, 2018). The qualitative approach provides a powerful tool for the researcher to analyze content, team dynamics, and processes through the narrative of individuals (Köhler, Smith, & Bhakoo, 2018). This study utilized a qualitative multiple case design to address the research questions. In the following section, I will establish the reasoning that undergirds the study's method and design choice.

Method

There are three types of research methods commonly employed in social sciences research: qualitative, quantitative, and mixed methods. Each has distinct features, benefits, and drawbacks. The qualitative approach provides experiences of the participants, an understanding of actions and events, and an interpretation of processes (Aagaard & Matthiesen, 2015). Researchers use a qualitative method to answer the question of "what," "how." or "why"' (McCusker & Gunaydin, 2015). A core strength of qualitative research is the variety of approaches it permits, the types of data that can be analyzed, the context of the data, and how it is treated or coded (Köhler et al., 2018). As this study explored strategies through the interpretation of the processes of the participants, I chose the qualitative method as the most appropriate approach as the objective of this research is to report on the “what,” “how,” and “why” of effective estimation strategies used by software development professionals. I selected a qualitative multiple case design to gain insight into effective estimation strategies that software development professionals use in providing accurate estimations of effort.

Although I considered other research methods, a qualitative methodology was the most appropriate choice. Qualitative methods use natural language, interpretation, and human expression as data for analysis and discovery of findings (Levitt et al., 2016). A qualitative researcher's goal is to provide a clear and vivid portrayal of phenomena through the gathering and development of data (Levitt et al., 2016). According to Collins and Stockton (2018) and Müller and Klein (2019), the qualitative research process begins with the identification of the problem or phenomenon. Following the identification, the researcher identifies relevant literature and determines a conceptual framework, participant selection, the role of the researcher, and an appropriate analytical process (Collins & Stockton, 2018). Finally, the researcher presents the findings and concludes with a discussion that relates to and answers the initial research question (Collins & Stockton, 2018). The qualitative approach provides an appropriate method to answer the question of this research as I used interviews to gather data followed by a qualitative analysis of responses as they may apply to the research questions.

The study I conducted engaged participants to uncover successful strategies for accurate estimation. Starcher et al. (2018) stated that qualitative research does not begin with a theory, but instead constructs meaning through an understanding of the phenomenon under exploration. One of the advantages of qualitative inquiry is that the methodology provides a tool to capture participant perceptions of the phenomenon, and these perceptions are the reality of the perceiver (Starcher et al., 2018). Levitt et al. (2016) summarized the qualitative process as the development of meaning via the researcher's reflection and the creation of conclusions from the meaning. Kelly (2017) states that the qualitative method is often exploratory to investigate the participants' opinions and viewpoints. I selected the qualitative approach as I conducted interviews to explore and understand strategies based on the evidence gained from the participant interviews. The interviews captured the participants' perceptions, realities, and a clear description of the processes and strategies that they use to provide accurate effort estimates.

An interview is a standard data collection process in qualitative research. It is uncommon for a qualitative researcher to conduct studies in a laboratory setting (Flannery, 2016). The researcher is the primary instrument in the data collection and interacts with the participant to construct an understanding through the gathered data (Starcher et al., 2018). The qualitative semi-structured interview method provides the researcher with tools to capture data in critical areas while still providing the flexibility to gain participants' personalities and perspectives (Barrett & Twycross, 2018). To understand effective strategies, I conducted semi-structured interviews to gain insight and understand effective strategies that the participants detailed. I scheduled and conducted the interviews in an environment that was comfortable, familiar, and convenient for participants.

I did not choose a quantitative method as the results of this research are inductive rather than deductive. Researchers select a quantitative method for the construction and identification of causally related entities, the establishment of correlations, and the utilization of numbers for the data material (Aagaard & Matthiesen, 2015). Quantitative research reaches conclusions deductively, whereas qualitative does so inductively (Kelly, 2017; Starcher et al., 2018). Researchers use the quantitative research method to provide statistical generalization (Carminati, 2018) and to express the research findings using numbers (McCusker & Gunaydin, 2015; Starcher et al., 2018). Quantitative methods differ from the qualitative methods, which answer the question of “how” and “why,” whereas the quantitative approach answers the question of “how many” and “how much” (McCusker & Gunaydin, 2015). To uncover effective strategies, I asked the participants the question of “how” and, thus, chose not to use a quantitative method as it would not be an appropriate method to answer my research question. The quantitative approach was not a viable option to answer my research question as the data is descriptive rather than numerical. Additionally, my research question could not be answered by a statistical generalization.

The purpose of this research was to uncover effective estimation strategies through an interview process, which did not involve causality or correlations. The research questions relating to effective estimation strategies required interviews for data collection. In quantitative research, researchers know what they are looking for, and the participants usually are kept separate from the researchers (McCusker & Gunaydin, 2015). The quantitative study begins with specific information and works towards a more general understanding to arrive eventually at a conclusion or explanation (Starcher et al., 2018). My research began with a question to explore effective strategies, and I did not have more than a general preconception of results. Additionally, I had direct contact with the participants of this study. Therefore, I elected not to use a quantitative method for my research as the qualitative method would not have provided conclusions to answer the research question.

A mixed-method approach combines quantitative and qualitative methods to analyze both narrative and numerical data (Venkatesh et al., 2016). The mixed-method approach is most commonly used by initially exploring the topic qualitatively, followed by a quantitative component, which is usually the primary research method (Green et al., 2015; McCusker & Gunaydin, 2015). The mixed-method approach combines a qualitative dimension to provide a deep meaning and a quantitative aspect to provide a statistical analysis (McCusker & Gunaydin, 2015). Although this study presents qualitative descriptions through evidence gained in an interview, there was no quantitative numerical analysis of the participant’s description of effective effort estimation strategies. Therefore, since my study was inductive and exploratory, and did not contain a numerical or statistical component, the mixed method would not be an appropriate choice.

Research Design

A qualitative researcher can select from multiple qualitative study designs: case study, ethnographical, narrative, and phenomenological. Although each study design has merit, I chose the multiple case as the most appropriate design to answer my research question. According to Stake (2006), researchers use multiple case designs to study phenomena in different environments. Awasthy (2015) posited that case studies cover the phenomenon and context of the characteristics of organizational processes. Additionally, the evidence in multiple case design is often considered more compelling and robust than a single case design (Yin, 2014). As this study will uncover effective strategies (processes) in different organizations (environments), I selected the multiple case study as my research design.

Case studies are an appropriate design when the phenomenon is broad, and a holistic, in-depth investigation is needed (Dasgupta, 2015). Ponelis (2015) asserts that the case study designs are useful in applied disciplines to study processes, problems, or programs to understand the phenomena and improve domain practice. Researchers use case studies to gatherer participant interpretations, report on their constructed reality or knowledge obtained through the investigation (Yazan, 2015). The case study design and other qualitative designs do not attempt to manipulate the phenomena or the study participants, but instead evaluate the results of naturally occurring activities or processes (Dasgupta, 2015). Dasgupta (2015) also claims that the researcher must study the phenomenon in the context in which it occurs. I selected the multiple case design as my study observes, analyzes, and interprets participant responses but did not manipulate or change them. My research examined various organizations (multiple cases) to identify effective software estimation strategies by investigating strategies and processes, as described by the participants.

I selected a multiple case design to gain insight and understanding of effective estimation strategies. For both single case and multiple case designs, researchers use observations, interviews, interpretation, and coding as the most common procedural elements (Stake, 2006). According to Dasgupta (2015), researchers use a multiple case design to study and identify similarities and differences across many instances. Researchers use case studies to focus on individuals' real-world perspectives regarding their home or work environment and their processes (Yin, 2014). Yin (2014) indicated that case studies provide answers to determine "how" and "why." Green et al. (2015) suggest that more resources and time are required in multiple case studies, but may offer more useful context in various sites. I used numerous participants and organizations to gain various perspectives to identify similarities and differences to processes and strategies in providing accurate software development efforts.

For this study, the process was to interview multiple participants from multiple organizations, interpret the results through a coding process, identify commonalities, and report on the findings from the data collected. The multiple case design provided data to understand and report on effective strategies in organizations' estimation processes. Awasthy (2015) indicated that multiple case designs uncover the phenomenon and the context of essential characteristics of organizational processes. According to Llerena, Rodriguez, Castro, and Acuña (2019), the multiple case study is useful for extending the information of the phenomenon, gathering more data than a single case, the examination of the phenomenon in numerous contexts, and addressing each case separately to describe conclusions for the research as a whole. Researchers use a multiple case design to draw conclusions based on similarities and differences across the cases (Dasgupta, 2015). I selected a multiple case design to identify estimation strategies from multiple organizations to conclude numerous perspectives. The multiple case design choice provided the data to gain an understanding of accurate estimation strategies, which makes the multiple case design the most appropriate option to answer my research question.

In an ethnographic design, researchers observe the behavior and culture of participants within a group. The ethnographic design provides a tool for the researcher to interpret a group's shared values and beliefs through observation in which the researchers themselves are immersed (Creswell & Poth, 2018). Moser and Korstjens (2018) state that the ethnographic study is a descriptive and narrative account of a specific culture. Ethnography is the study of groups of people and cultures most commonly used by researchers in anthropological studies (Awasthy, 2015). This study did not investigate the culture of the participants or their shared values and beliefs. I focused the study on participants who are engaged in a professional domain and not selected based on their culture. Therefore, the ethnographic design would not be appropriate for addressing the research question of this study.

The narrative design describes a story or explores the life of a participant. Creswell and Poth (2018) and McAlpine (2016) agree that a narrative design is used to collect stories and lived experiences of an individual. In the narrative inquiry, participants are encouraged to tell a story about their lives through dialogue with the researcher (Barrett & Twycross, 2018). The focus of this study was to uncover strategies that required an understanding of the participants' processes and perspectives rather than their individual stories. This study is not concerned with the life stories or lived experiences of the participants, but rather with accurate and practical strategies that the participants use to estimate effort in software development. Thus, using a narrative approach would not answer my research question; therefore, I did not select it as the design.

The phenomenological design describes the ordinary meaning of the experiences of several individuals. Although the participants' interpretation is essential to answering the research question, I did not choose the phenomenological design for this study. The phenomenological design expresses the lived experiences of a common phenomenon of the participants and the interpretation of the lives that they lead (Alase, 2017; Ellis, 2016). The purpose of the study was to discover effective strategies and not how the participants experience daily life. The phenomenological approach differs from the case study design as a case study approach uses themes and categories. In contrast, the phenomenological design tells a story through the lived experiences of the participants (Alase, 2017). A phenomenological study describes a collective experience of the participants in sharing phenomena and concludes with the essence of “what” and “how” the participants experienced it (Creswell & Poth, 2018). Although my research question was answered by the participants describing “what” and “how” of estimation strategies, I did not use the phenomenological design approach as it was not the intention to understand the shared or lived experiences of the participants, but instead to understand the strategies used by the participants in providing accurate estimates.

Moser and Korstjens (2018), and Malterud, Siersma, and Guassora (2016) stated that the researcher obtains data saturation when no new analytical information is discovered. Boddy (2016) asserts that data saturation can only be achieved with two or more cases, as one single case is never enough. Boddy (2016) further states that a researcher can achieve data saturation with as little as six in-depth interviews. According to Fusch et al. (2018), the use of multiple sources of data enhances data saturation. The multiple case design was selected to explore and analyze effective estimation strategies from multiple individuals in various groups to understand the differences and similarities of the estimation strategies. To achieve saturation, I selected five organizations and interviewed two individuals from each organization to achieve data saturation. Additionally, I reviewed documents from each organization that provided data on estimation strategies.

Population and Sampling

The population selected for this research was software development professionals from multiple teams in multiple organizations located in South Texas. South Texas has three major cities, San Antonio, Austin, and Houston, each of which has many companies that employ internal teams to develop software for internal use. The United States Department of Labor reports that over 32,000 application software developers worked within San Antonio, Austin, and Houston areas in 2018 ("Bureau of Labor Statistics - Software Developers, Application," 2019). In conducting research, it is essential to identify participants who can provide depth, breadth, and quality of data for accurate analysis and reporting (Saunders & Townsend, 2016). Determining the sample size in qualitative research is based on the researcher's judgment, as too large a sample can affect the depth of a study, and too small of a sample may not produce data saturation ((Boddy, 2016; Carminati, 2018). Boddy (2016) states that a sample size of six can be used to reach data saturation provided the researcher conducts in-depth interviews. Yin (2014) indicates that six to ten cases are sufficient to produce compelling evidence. The sample size of this study was 10 participants from multiple different development teams. Each participant had at least five years' experience in estimating software development effort. Additional criteria for inclusion in this study are that the participants are currently in an active role in software development, the product that is under development requires estimating, and the participant has the knowledge of and is currently working with strategies that are effective in providing accurate estimates. This study is not trying to achieve certainty, but rather, it is exploratory.

There are two primary types of sampling methods; probability and nonprobability (Rahi, 2017; Sarstedt, Bengart, Shaltoni, & Lehmann, 2018). Although probability sampling is a viable method for establishing a representation of a population (McCusker & Gunaydin, 2015; Sarstedt et al., 2018), I selected a nonprobability approach. Nonprobability sampling is a non-randomized intentional selection of participants based on subjective methods in the inclusion decision (Etikan, Musa, & Alkassim, 2016). A distinction of the probability sample approach is that each person has an equal chance of inclusion in the study (Rahi, 2017). My study reports solely on strategies that provide accurate effort estimates rather than all strategies. Thus, I have chosen a nonprobability method as the best method for this research as I selected participants based on a predetermination that the participants meet the criteria of knowing and using an accurate estimation strategy.

I have chosen the judgment or purposeful nonprobability method for this study. The judgment or purposeful approach allows researchers to use their judgment in selecting the participants (Rahi, 2017). The purposeful sampling selection method is based on the researcher's assumed judgment and expertise to select participants who are deemed appropriate and will provide data for analysis of the effect under study (Sarstedt et al., 2018). Purposeful sampling is the selection and intentional inclusion of participants with the knowledge and experience to assist in the analysis and interpretation process (Twining, Heller, Nussbaum, & Tsai, 2017). Tong and Dew (2016) state that a purposive sampling strategy is a deliberate choice of participants who can articulate perspectives pertinent to the research question. The purposeful sampling method involves selecting participants based on the knowledge of the researcher (Wilson, 2016). Before including a participant in the study, using a purposeful selection method, I conducted preliminary interviews (see Appendix A) to establish that the participants had knowledge of and are currently using an effective and accurate estimation strategy. The participants have to answer “yes” to all the preliminary questions listed in Appendix A to confirm that they align with my research question for inclusion in the study. As I have a background in software development and estimating the effort required to complete various goals, I also relied on my professional judgment to confirm the use of a successful estimation strategy.

The purpose of this study was to gain an understanding of effective and accurate strategies in software development effort estimation. I purposely selected participants that meet the qualification for inclusion to this study. Sarstedt et al. (2018) state that purposeful sampling is an appropriate selection method when the researcher analyzes results for an improved understanding rather than for the generalizability of results. Purposeful or judgment sampling is the deliberate choice of participants based on criteria or qualities the subjects possess (Etikan et al., 2016). Purposeful sampling is the selection of individuals who are well informed about the phenomena of interest and can communicate their experiences in a reflective manner (Etikan et al., 2016). I selected the purposeful expert sampling approach, as I used individuals who have a unique knowledge of estimation and currently use effective strategies. As the name implies, expert sampling is the selection of subject matter experts who have previous experience of the subject matter of the study ((Etikan et al., 2016). Purposeful sampling allows the researcher to select participants to obtain a comprehensive understanding of the phenomenon with the expectation that each participant will provide substantial information to the study (Etikan et al., 2016).

Additionally, purposeful sampling is selecting a limited number of participants who can provide an in-depth understanding of the phenomena for the researcher to report conclusions (Yilmaz, 2013). The participants chosen for this study were chosen based on the prequalification that each understands software estimation and uses accurate estimation strategies. I ensured that the participants meet the qualifications before interviewing to minimize the sample size while providing conclusive results using a limited number of individuals. For this study, I interviewed ten qualified participants. According to Etikan et al. (2016), nonrandom purposive sampling does not need a set number of participants. Malterud et al. (2016) suggested that small sample sizes can be sufficient if (a) the aim of the study is narrow, (b) the characteristics of the participants are highly specific, (c) the researcher has a theoretical background, and (d) the researcher maintains active communication with the participants. However, the sample size must be evaluated throughout the research (Malterud et al., 2016). Carminati (2018) states that the sample size is essential to generalization, as too large a sample inhibits in-depth analysis while too small a sample does not support saturation or redundancy. A researcher cannot be sure the chosen sample is generalizable in nonprobability sampling (Wilson, 2016). Qualitative studies should consider the strength of the information and knowledge gained from the analysis rather than putting a strong emphasis on sample size (Malterud et al., 2016). The selection of ten participants provided satisfactory results, as the research question is narrow.

I carefully selected the participants based on predetermined criteria and will conduct an in-depth interview to collect a full perspective from each participant. This study did not attempt to generalize but instead provided me with insight into the usage and processes of effective and accurate estimation strategies. Additionally, I evaluated the sample size throughout the investigation to ensure that the data gathered provided insight and answered the research question adequately.

Ethical Research

Before I collected data from the participants of this study, I gained the approval of the Walden University Institutional Review Board (IRB). The primary purpose of an IRB is to review the protocols and processes of the research to ensure that no harm comes to the participants, and sufficient measures are in place to minimize risk (Miracle, 2016). Alase (2017) further detailed the responsibility of the IRB to ensure that any devices, techniques, or strategies the researcher uses have the full consent of the research participants, and the IRB has approved for use. Additionally, the IRB determines if the risks and benefits are balanced, the recruitment strategies are fair, and the researcher has sought voluntary consent (Bracken-Roche, Bell, Macdonald, & Racine, 2017). This study met all legal and ethical requirements established by the Walden University IRB. The Walden University's approval number for this study is 12-19-19-0421147.

In addition to obtaining the Walden University IRB approval, I adhered to the standards outlined in the Belmont Report. Ethical research follows principles that the researcher should follow to protect the participants from harm. The Belmont Report defines ethical principles as respect for persons, benevolence, and justice (The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research, 1979). Miracle (2016) described the three principles of the Belmont Report as a guide for researchers in (a) respecting that people have the right to decide whether they want to participate, (b) doing no harm, and (d) treating all participants equally.

Throughout this study, I ensured that no harm came to the participants by taking all safeguards regarding confidentiality. I provided equitable and fair treatment for persons and organizations involved in this study. Additionally, I reminded each interview participant that the study's involvement was voluntary and that there was no financial or professional compensation for participation in the research. I informed each participant that they could withdraw from the study before the analysis phase of the study by contacting me through e-mail or phone.

All the participants in the study consented both verbally and by signing a consent form. Brown and Danaher (2019) state that the consent form should be understandable, informative, and clear to the participant without being vague. Miracle (2016) suggests that a consent form should include the following components: a) purpose of the study, b) description of the research procedures, c) potential risks and potential benefits of the study, and d) an indication that participation is voluntary. Arsel (2017) and Ponelis (2015) suggest that the interview should include a preliminary discussion to establish informed consent and that the researcher should inform the participant of any consequences of participation in addition to obtaining the participant’s signature on the consent form. The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research state in the Belmont Report that the informed consent should include a statement informing the participant that they may withdraw from the study (The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research, 1979). Before I began the interview, I discussed the study purpose with each participant, any potential risks, and reminded them that participation is voluntary. I verified that they understand the consent form and reiterate that they may withdraw from the study at any time during the interview or through e-mail correspondence or phone call before to the analysis of the data.

Arsel (2017) suggests requesting a signed consent form to ensure the participant understands the procedures and consequences of participation in the study to avoid any misunderstanding of involvement in the research. Miracle (2016) states that an informed consent document should include the purpose of the study, identification of procedures and risks, and a notice to the individual that participation is voluntary. The Belmont Report identifies three necessary items for informed consent: (a) the inclusion of information about the study; (b) the participants should fully comprehend the consent; and (c) the participants should understand that participation is voluntary (The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research, 1979). I requested that all the participants sign a consent form before the interview and verbally verify that they understood the consent, procedures for withdrawal and that they will receive no compensation for involvement in the study.

Throughout this study, I took precautions to preserve the privacy of the participants and the organizations that employ them. All information regarding any indication of the identity of any participant or any organization will be held and stored in an encrypted folder for five years. I am the only person who will know the identity of the participants and organizations in this study. At the end of the five years, I will delete the contents of the folder, and destroy all hard copy data about the individuals or organizations used in the study. Any data that I publish will not include any personally identifiable information or information that would identify the organizations that employ them. The published study will only use codes or pseudonyms to identify the participants or organizations. Using pseudonyms for both the participants and the organization is a common practice in research (Allen & Wiles, 2016). Data indicating the participants' gender, race, or any other information that would disclose the participant's identity should not be used in the study if that information is not relevant for the study (Allen & Wiles, 2016). The study does not include names or the organization that employs them, but, instead, I identified them using pseudonyms such as participant 1, participant 2 and so on. Additionally, I will not publish any information that would indicate race, gender, or age.

Before beginning the interviews, I informed each participant of the purpose of the study, that participation is voluntary, procedures for withdrawing, and that I will safeguard their identities. Arsel (2017) maintained two tenets for interviewing participants: to ensure that the participants understand that the answers to the questions, and for research and do no harm. I verbally discussed with each participant in the study the intent of the research, their right to opt-out of the study voluntarily, and their right to confidentiality under all circumstances. I informed each participant that there would not be any financial or professional compensation for participation. Additionally, I ensured that each participant fully understood and agreed to the involvement in the study and provided them with a copy of the consent form.

I informed each participant that the data collected will not be used for any purpose other than the study. According to Starcher et al. (2018), recorded interviews should be transcribed verbatim to provide the researcher with analysis for a credible understanding of the studied phenomenon. I will hold the interview tapes and transcripts for five years in a secured digital container, and no personal information on the data will identify the interviewees or their responses. After five years, I will delete all the interview records, recordings, and transcripts. Additionally, after five years, I will shred all field notes from this study and discard them.

Data Collection

In a qualitative study, researchers can collect data from many sources. Yin (2014) indicated that traditional sources of evidence are documentation, archival records, interviews, direct observations, participant observations, and physical artifacts. Qualitative research relies on three sources for data collection: observation, interviews, and documents (Kelly, 2017; Starcher et al., 2018). In the following sections, I describe the instrument I used, my data collection technique, and the method I employed for data organization.

Data Collection Instruments

I was the primary data collection instrument for this qualitative multiple case study. Starcher et al. (2018) and Babchuk (2019) state that in qualitative inquiry, the researcher is the primary data collection instrument. As I was the primary data collection instrument, I followed the interview protocol found in Appendix C to maintain consistency across the interviews that I conducted. Ismail et al. (2018) state that the utilization of an interview protocol increases the interview process's efficacy by ensuring the researcher attains comprehensive data within the allocated time. Ismail et al. (2018) further suggest that the researcher conduct a pilot interview to check the effectiveness of the interview protocol. Conducting a pilot interview provides the researcher with a crucial test of the interview questions and gives the researcher interviewing practice (Majid, Othman, Mohamad, Lim, & Yusof, 2018). I conducted a pilot interview to gauge the effectiveness of my interview protocol. Additionally, my pilot interview provided me with an indicator of the adequacy and completeness of the responses I would receive from the research participants.

The primary data collection method for this study was semi-structured interviews. In semi-structured interviews, participants are free to respond to open-ended questions as they wish, and the researcher can ask supporting questions to explore deeper into the participant's reasoning (McIntosh & Morse, 2015). Deterding and Waters (2018) define semi-structured interviews as open-ended questions that generally follow a logical order designed to create a dialog between the researcher and the participant. Open-ended research questions allow the responder to provide an answer that makes sense to them (Windsong, 2018). The principles of semi structured interviews are that the method helps a researcher to stay on topic, to construct data, and to guide the discussion (Starcher et al., 2018). I selected the semi-structured interview data collection method as it allows the freedom to gain a deeper understanding of the participants perspective using follow-up questions while still maintaining a structure for the interview process. Additionally, the semi structured approach provides the participants with the opportunity to answer the interview questions based on their perspectives.

I asked the participants semi-structured interview questions to explore the estimation strategies use by the participants as detailed in my interview protocol (see Appendix C). I used semi-structured interviews with pre-selected participants to produce the data in this study. The semi-structured interview process provides the participant’s perspective on the research topic (McIntosh & Morse, 2015). Using the semi-structured interview process, I asked each participant the same questions in the same order to ensure consistency in the data collection interviews with the participants. I selected the semi-structured methods to provide a structure for the interview process while giving me the freedom to gain a more in-depth insight using additional nonpredetermined questions.

To enhance the study's reliability and validity, I reviewed company documents to supplement and support the information gained in the interview process. Brooks and Normore (2015) state that documents contain information previously established outside of the researcher's intervention and adds rigor to the study. Triangulation of the data, such as interview data and records or documents, help the researcher understand the circumstances (Ismail et al., 2018). Researchers achieve triangulation through participant interviews, document analysis, and direct observation (Babchuk, 2019; Fusch et al., 2018). Once I completed the interview with the participant, I asked permission to examine any company documents related to the research topic to confirm the estimation strategy identified by the participant. In addition to participant interviews and document analysis, I collected field notes to enhance my data collection process.

Throughout the research process, I maintained a reflective journal and field notes to capture personal thoughts, interpretations, and observations to aid in the documentation and analysis of contextual information. According to Zulfikar and Mujiburrahman (2018), a reflective journal is used to organize thinking and provide self-evaluation opportunities. Researchers use reflective journals to examine their responses to participants, consider the interview questions' effectiveness, and evaluate their responses to the data they collect (Orange, 2016). Phillippi and Lauderdale (2018) suggest that interview field notes should include information regarding the setting, the overall demeanor of the participants noting any nonverbal behaviors, any deviations in the interview process, critical reflection after the interview, and notes regarding the interviewer's self-assessment regarding performance. Additionally, I continuously captured and reflected on the data collection process, interview approach, and personal interpretations to counter any potential bias. The field notes provided an additional data source, a more in-depth analysis, and a record of my interpretation of the findings.

Data Collection Technique

Once I received approval from the Walden University IRB, I conducted a pilot interview to confirm the adequacy of my interview protocol (see Appendix C), and I began the data collection process. The data collection process consisted of interviews and document analysis. The most common data collection methods in qualitative research are participant observations, interviews, and focus groups (Moser & Korstjens, 2018). The interview technique followed the protocol detailed in Appendix C, and I ensured that the location and setting were comfortable for the participants. The location of the study can affect the outcome of the study (Rimando, Brace, Namageyo-Funa, Parr, & Sealy, 2015). The first few minutes of an interview are critical for allowing the participant to be at ease and to feel that they can freely discuss the topic and tell their own experiences (Moser & Korstjens, 2018). I began the interview by introducing myself, reviewing the signed consent forms with the participants, and providing the participants with an opportunity to ask questions or voice any concerns before the interview begins. Before beginning the interview questions, I ensured sure the participants were comfortable, receptive, and ready to be interviewed.

There are advantages and disadvantages of using a semi-structured interview approach and analysis of organizational documents. The semi-structured interview process provides the researcher with a method to obtain the participants' perspective and experience regarding the research topic (McIntosh & Morse, 2015). The semi-structured interview allows the researcher to follow a listing of questions during an interview while enabling the interviewer the opportunity to elicit open responses to develop a deeper understanding of the perspective of the participant (Brown & Danaher, 2019). However, according to McIntosh and Morse (2015), the disadvantages of face-to-face interviews are participants may feel inhibited when asked to respond to sensitive questions. Additionally, an interviewer's physical presence may affect the participant's response, and conducting interviews is time-consuming and costly (McIntosh & Morse, 2015).

Document analysis is the reviewing or evaluation of documents. The combination of interviews and document analysis contributes to the rigor of the research (Fusch et al., 2018; Yilmaz, 2013). Organizational documents contain information developed without the researcher's participation or intervention (Brooks & Normore, 2015). However, Brooks and Normore (2015) indicate that documents can present individuals' or organizations' perspectives and may not represent the participants' perspective. For this study, I analyzed each organization's documents to support the data obtained in the interview process.

I explained to each participant that I would record the interview and transcribe the information to use for data analysis for the study. Recording the interview provides a method for the researcher to capture the information for later transcription (McGrath et al., 2018). Researchers use audio recordings to transcribe interviews verbatim for analysis (Cypress, 2017; McGrath et al., 2018; Starcher et al., 2018). Following any questions that the participants may have, I started the formal interview process. I began the interview process by turning on my recording device, stating the date and identifying the participant as participant one, two, three, and so on.

I reminded the participants that the recording, personal notes, and the transcription will not include any personally identifiable information and that I will maintain their anonymity and preserve confidentiality. Researchers should safeguard participants' responses to ensure that the published results do not disclose their identity or put them in a vulnerable situation (Arsel, 2017). The interviewer's primary task is to understand the meaning of the participant responses (Korstjens & Moser, 2018). The study was constructed to provide successful strategies used for effort estimation while not disclosing any information that would identify the participants or organizations. Additionally, it is essential to communicate to the participant how the researcher will maintain privacy and confidentiality as the level of trust between the researcher and participant affects the quality of results from an interview. (Brown & Danaher, 2019). I made the participants aware that I would ensure that personal information remains private and that any information discussed would not be shared with their organization, supervisor, or coworkers.

Interviewers should demonstrate that the responses the participant provides during the interview are understood. Positive responses acknowledge that the interviewer understands the responses through body language, response tokens, and the formulation and asking of the next questions (Roulston, 2018). According to Roulston (2018), body language and response tokens such as an interviewer interjecting “yes,” and “I am following” provide the interviewee with feedback indicating the interviewer understands the dialog. To gain additional information or to prompt a participant to expand on an answer, researchers can interject “tell me more about that” after the participant's response (Starcher et al., 2018). Fusch et al. (2018) state that follow-up and probing questions help the researcher maintain the direction of the interview and collect additional data to answer the research question. Roulston (2018) suggests repeating back to the interviewee the interviewer’s understanding as well as asking the next questions will provide the participant with a positive affirmation that the responses are understood. During the interview process, I ensured that participants understood the questions and that I understood their responses. I asked follow-up questions if I determined that a more vibrant response was required or that the participant did not answer the questions adequately enough for me to gain an understanding.

I took field notes during the interviews. According to Barrett and Twycross (2018), field notes include a chronological log, an account of what the researcher observes, and an expanded interpretation of impressions from the interview. Field notes are a widely used approach to capturing contextual information and use in subsequent analyses and synthesis of the data collected (Phillippi & Lauderdale, 2018). Interviews and field notes are the principal data sources in qualitative research (Moser & Korstjens, 2018). I asked the interview questions in the exact order outlined in the interview protocol and allowed the participant to respond adequately before asking follow-up questions. I continued the interviews until the participants had answered all the questions.

I limited the research questions such that the questions asked would elicit a response that answered my research question. Data collection can be adversely affected by the length of the interview process (Rimando et al., 2015). I completed the interviews within one hour. Following the interview questions, I asked the participants if there was any information they would like to share or any information they feel should be included in the study. I documented the additional participant comments and information that was relative to my research question in the field notes.

I asked the participants if there was any company documentation that they can share relevant to the topic discussed. Chung (2019) states that the reliability of documentary evidence is higher than that of verbal evidence. Yin (2014), as well as Creswell and Poth (2018), indicate that the use of documents is to corroborate and supplement interview data with other sources of data. After conducting the interviews, I request to see any documentation such as process and procedures documents, standard operating procedures, checklists, or guidelines used in estimating effort. I reviewed the documentation to ensure alignment with the participant’s responses and discuss any deviation that I identify. I notated my document review observations in my field notes.

I explained the concept of member checking and informed each of the participants that I would be contacting them by phone to discuss my interpretations of their interview responses. The member checking process is critical to verify the accuracy of the content and understanding of the participant’s viewpoints (Candela, 2019; Yilmaz, 2013). Creswell and Poth (2018) define member checking as reviewing with the participant the accuracy of the researcher’s interpretation findings and as verifying that the participant answers are representative of their intended responses. Yin (2014) describes member checking as the corroboration of interpretations of the interview with the participant and to allow for new evidence to emerge not gained through the initial data collection. I requested time for a brief phone discussion of the interview to check for the accuracy of my interpretation of the interview and supporting documents. Participant verification provides a tool for member checking and allows the participant to confirm the dialog transcribed during the interview is as the participant intended. If the participant indicated that a change in a previous response was needed, I made the change and notated the change in my field notes. Additionally, I scheduled a time to contact the participant for review and confirmation of the amended material.

I thanked each participant and confirmed that each participant had my contact number and e-mail should they have any questions or remember any additional information that they have not previously discussed that was relevant to the research topic. Following each interview, I transcribed the audio recording into separate Microsoft Word documents. Transcribing an interview verbatim provides the researcher with a credible understanding of the phenomenon studied (Starcher et al., 2018). I removed any information that would identify the participant, company, or development team members to ensure confidentiality.

Data Organization Techniques

The data organization approach that I used for this study was to store recordings, field notes, and transcribed data on a secure folder on a personal secure Microsoft OneDrive cloud storage account. Babchuk (2019) stresses the importance of organizing data in a meaningful way and storing the data on a password-protected computer. Van Baalen (2018) suggests storing data on encrypted folders using strong passwords for digital security. I am the only person who has access to the protected folder, thus providing safeguards to protect the confidentiality of the participants and organizations. Surmiak (2018) defined research participants' confidentiality as the nondisclosure of participant identifiable information unless they consent to the disclosure. I stored each participant interview recording, notes, and audio transcription in separate digital folders, indicating a unique identifier such as participant 1, participant 2, and so on to provided confidentiality. I stored each of the participant folders in a digital folder indicating their organizations such as organization 1, organization 2, ensuring the confidentiality of the organizations. I labeled each file associated with each participant to indicated participant number, organization number, and research artifact types, such as a transcript, recording, and note.

I documented my feelings, understanding, and personal thoughts throughout the research project using a reflective journal. Levitt et al. (2016) suggested the use of a reflective journal to manage perspective. A reflective journal includes a researcher's emotions, beliefs, and reasoning inferences (Bruno & Dell'Aversana, 2017). Researchers use reflective journals to promote validity, promote learning through the research process, and provide evidence of transparency (Vicary, Young, & Hicks, 2017). Additionally, a reflective journal allowed me to review thoughts and observations that are otherwise not documented. A reflective journal provided a resource for the research progression, observations, and personal reflections, thoughts, and feelings throughout the process.

I stored my field notes and reflective journal in a locked cabinet to ensure that only I can access the files. All the artifacts of the research are available only to me and protected from unauthorized access. I will store the participants' recordings, transcriptions, field notes, and encrypted identification information for five years. After five years, I will delete all participant information, recordings, transcripts, and shred field notes and reflective journals.

Data Analysis Technique

To answer my study question, I repeatedly searched the data I have collected until I achieved a meaningful answer to identify effective strategies in software development effort estimation. Tong and Dew (2016) state that the data analysis process is iterative and revelatory, and generally involves examining the data, categorizing and grouping similar concepts into themes to identify relationships and patterns. Babchuk (2019) states that the analysis should include verbatim phrases taken from the participants to capture the connotation of the line or text passage. For this study, I found meaningful information through the analysis of data collected from interviews and organizational documents related to effective strategies to estimate software development effort. I analyzed the data, derived codes, and identified themes to discover effective estimation strategies in software development.

According to Creswell and Poth (2018), a typical analysis approach for multiple case studies is to conduct a within-case analysis of individual cases, followed by a thematic cross case-analysis across multiple cases. A cross-case analysis is the examination of themes across cases to identify similarities and differences (Creswell & Poth, 2018). Yin (2014) defines cross-case analysis as the aggregation of findings across multiple cases. Researchers use a cross-case analysis to gain an understanding of common and unique features of multiple cases (Guetterman & Fetters, 2018). A cross-case analysis enhances transferability and trustworthiness by comparing data across multiple cases. I analyzed each case to identify themes and concepts followed by a cross-case analysis to identify similarities and differences to understand the phenomenon in a different context. I repeatedly reviewed my data to discover meaningful information to answer my research question. My data analysis focused on the discovery of similarities, differences, and correlation of effective strategies in software development effort estimation.

I used methodological triangulation to analyze the data gained from company documents and participant interviews. Abdalla, Oliveira, Azevedo, and Gonzalez (2018) state that researchers use the methodological triangulation to obtain complete and detailed data by analyzing multiple data sources such as interviews, observations, and documents to understand a phenomenon. Researchers use the methodological triangulation process to avoid bias and view data from multiple perspectives (Fusch et al., 2018). Methodological triangulation overcomes weakness and bias that result from single method research, single data, single observer, and single theory research (Joslin & Müller, 2016). As my study is qualitative only (single method), I was the only interviewer (single observer), and my conceptual framework is the planning fallacy (single theory.) The use of the methodological triangulation adds validity thru a sound approach to my data analysis. I used multiple sources to provide data for this study, such as interviews, organizational documents, and participant observations during the interview process.

The initial step in my data analysis was to review my research data repeatedly. I captured my thoughts in a reflective journal during my review process and made a note of any relevant themes, concepts, and similar or contrasting content. In analyzing qualitative data, the first step is to derive codes and identify essential words or phrases (McIntosh & Morse, 2015). Constructing categories or themes by grouping similar or closely related codes is the process of identifying similarities ((Babchuk, 2019). I repeated this process to identify meaningful information that was relevant to answering my research question. Additionally, I incorporate any new studies relevant to my findings after my proposal was accepted and before drawing my research conclusions.

Deterding and Waters (2018) suggested the first step in examining the data in qualitative analysis is that the researcher should identify the main themes to determine a provisional idea of the emerging themes and explore themes indicated by previous literature. The second step is to note specific chunks of text in which the participant was particularly articulate and concise (Deterding & Waters, 2018). The third step is to use qualitative data analysis software to explore the depth of the story, identify trends, and analyze potential negative cases that may limit the explanation (Deterding & Waters, 2018). Reducing data from full transcripts to indexed extracts and finally to grouped analytic codes provides the researcher with uniformity and increases reliability and validity (Deterding & Waters, 2018). I followed the process of establishing themes, exploring relationships, identifying trends, and using induction to report the research findings and conclusion.

I used NVivo, a Computer Assisted Qualitative Data Analysis Software (CAQDAS) application for the data analysis process. The primary advantages of using a CAQDAS software product are increased speed in handling large amounts of data, improved rigor, the identification of counts of phenomena, search for divergent cases, and the development of coding schemes (Cypress, 2019). Researchers use NVivo to reduce the workload in analyzing and structuring large amounts of data, searching for words or phrases, and to apply assigned codes to the text (Røddesnes, Faber, & Jensen, 2019). NVivo provides multiple qualitative analysis functions such as sorting, filtering, assigning, and defining categories themes as well as data visualization (Phillips & Lu, 2018). I used NVivo to import each participant's data, create nodes to develop a hierarchy to identify data between cases and within cases, conduct data exploration using the query command for similarities, matches, word frequency, and establish text patterns and keywords for code creation.

Reliability and Validity

Reliability and validity are foundational elements of proper research. Cypress (2017) defined reliability as a principal factor of research reflected in the practices, process, analysis, and conclusions. Validity is the state in which the research is grounded, justifiable, relevant, meaningful, and conforming to quality principles (Cypress, 2017). There are four quality attributes for establishing validity and reliability for a qualitative study. A researcher establishes quality and reliability by dependability, credibility, transferability, and confirmability (Moser & Korstjens, 2018).

Credibility

Credibility is the quality of trust and believability of research and its internal validity determined by the plausibility of the information and interpretation of the views of the participants (Moser & Korstjens, 2018). Researchers establish credibility through triangulation and member checking (Moser & Korstjens, 2018; Tong & Dew, 2016). Cypress (2017) states that researchers obtain credibility through the accurate and truthful description of the participant’s experience, and member checking, which is the constant checking of data and interpretations with the participants interviewed. Additionally, Moser and Korstjens (2018) defined credibility as an accurate description of the phenomenon and generation of believable claims through the identification of the study design, sampling method, data collection methods, identification of limitations and delimitations, and reflexivity. Kelly (2017) states that qualitative research's credibility is gained through good quality interviewing procedures, accurate coding and analysis, transparent conclusions, and evidence that the reader can transfer to their situations.

Research is reliant on the researcher's ability and effort, and descriptive narratives of the participants to achieve credibility (Cypress, 2017). I used member checking and methodological triangulation to establish credibility for this study. Additionally, I ensured that any claims made were free of bias and were an accurate account of each participants' viewpoints.

Researchers use member checking to ensure the reliability of their research. Member checking provides participants with an opportunity to correct misrepresentation or errors in the information gained during the interviews (Tong & Dew, 2016). Member checking is a commonly used procedure to share with the participant the data from their interview and the interpretations of the researcher to obtain the participant's feedback (Liao & Hitchcock, 2018). Candela (2019) stated that an additional benefit of member checking is that it helps the researcher capture the voice of the participant. After each interview, I transcribed the conversation and verified my findings with each participant to confirm correctness. Additionally, I discussed with the participants, my understanding of the strategies discussed to verify that information obtained from the interview dialog was accurate, and as the participants intended. I ensured methodological triangulation using interview transcripts, the review of organizational documents, and details noted during the interview process.

Transferability

Research transferability is the degree to which other researchers can transfer the results to other contexts with different respondents (Brooks & Normore, 2015; Moser & Korstjens, 2018), and other researchers can transfer the results to other settings (Shannon-

Baker, 2016; Tong & Dew, 2016). Researchers establish transferability by providing full descriptions of the participant's experience, context, and behavior (Moser & Korstjens, 2018). I captured detailed information from the study and a descriptive analysis of the research experience to validate transferability. Additionally, I documented the context, the interview setting, the participants' descriptions, their nonverbal behaviors observed during this research, and any other information that may help other researchers replicate or extend the study.

Dependability

I used member checking and triangulation to ensure dependability for my study. Xerri (2017) has stated that dependability and creditability increase through member checking and triangulation. Dependability is the stability of the study's findings over time and the fidelity of the data received from the study (Moser & Korstjens, 2018). Dependability is the consistency across the research methodology, data collection, and reporting of results to include transparency and verification of the research process (Tong & Dew, 2016). According to Lishner (2015), research trustworthiness and dependability increases when the researcher (a) promotes direct replication studies, (b) shares data when requested, and (c) adopts a truth-seeking mindset during the research process. I used member checking to ensure that my understanding and interpretation of the interview data was accurate. In addition to member checking, I triangulated the interview data with company documents when available to confirm the participant's strategies. Additionally, I adhered to my interview protocol to maintain consistency through my interview process. I kept a reflective journal detailing my data collection process, my thoughts on data analysis, observations during the interview, and personal reflections throughout the research process.

Confirmability

Moser and Korstjens (2018) and Korstjens and Moser (2018) noted that research confirmability involves confirmation of a study’s findings by independent researchers. Researchers establish confirmability and dependability by maintaining a reflexive journal to document thoughts and research notes that create an audit trail to document the collection, analysis, and interpretation of the data (Cypress, 2017). The study's results are grounded in data, and I ensured that my viewpoints and biases were identified and mitigated. Additionally, I maintained a reflective journal to capture personal thoughts and feelings during the interview process. I used NVivo to identify reoccurring themes to indicate data saturation.

I used a semistructured interview with multiple members from multiple teams to gain data saturation. Fusch et al. (2018) state that the use of multiple sources of data enhances data saturation. According to Abdalla et al. (2018), researchers attain confirmability by ensuring that the conclusion drawn from the interviews comes from the experiences and ideas of the respondents. I provided member checking to promote confirmability. Additionally, throughout the research process, I remained transparent about my approach and findings.

Transition and Summary

In Section 2, I presented details of the research plan, my role as a researcher, the research design, and the data collection techniques for this study. Additionally, I have provided processes to ensure reliability and to validate the study. Section 3 will include the research findings, implications for social change, suggestions for professional practice, and recommendations for future research followed by personal reflections on the research and my conclusions.

Section 3: Application to Professional Practice and Implications for Change

The focus of this study was on exploring strategies used by software development professionals in providing effort estimations. This section includes (a) an overview of the study, (b) presentation of the findings, discussion of the study's (c) application to professional practice and (d) implications for social change, (e) recommendations for actions, (f) suggestions for further study, and (g) personal reflections and a study conclusion.

Overview of the Study

The purpose of this qualitative multiple case study was to explore strategies used by software development professionals to provide project managers and product stakeholders with accurate effort estimates. The data came from interviews and documents within five different organizations located in South Texas. All the participants interviewed were actively involved in providing estimations. Each participant had at least five years of experience in delivering software development effort estimates, and each indicated that the strategies used were effective. My analysis of the data resulted in four themes that were common among the participants for achieving accurate effort estimations. Although the methods in providing estimates differed within the teams, the participants' strategies to arrive at accurate estimates were common among the participants.

Presentation of the Findings

The main research question for this study was as follows: What are the strategies that agile software development teams use successfully to provide their project managers with accurate estimates of software development effort? This section includes a summary and analysis of the results of the interviews and organizational documents identified, resulting in four main themes related to providing accurate software development effort estimates. Access to organizational documents allowed me to triangulate and validate the information obtained from the interviews. I conducted the interviews such that participants had the opportunity to share strategies that they found to be essential in providing estimations of effort in software development activities. After transcribing the interviews from all 10 participants, I imported the transcriptions into NVivo for analysis and coding. I imported the organization documents into NVivo for analysis and coding. Four emergent themes resulted from my analysis: (a) define and decompose requirements; (b) reference historical data; (c) identify risks and unknowns; and (d) foster communication, collaborations, and consensus. The participants discussed their strategies for helping project managers to plan delivery schedules, thus improving customer satisfaction. Additionally, accurate estimations can also enhance the quality of the product, lower stress levels, and improve the work lives of those involved with software development and delivery (Yamini & Marathe, 2018).

The planning fallacy (Kahneman & Tversky, 1977) was the conceptual framework for the study. It is common for developers to underestimate effort estimations, which adversely affects planning, delivery schedules, and cost (Løhre & Jørgensen, 2016). The themes identified provide strategies to mitigate the effects of the planning fallacy, potentially resulting in predictable schedules and higher accuracy in the planning of software development delivery.

Theme 1: Define and Decompose Requirements

Defining and decomposing requirements was one of the prominent themes. The process of requirements definition is that the requested work item is sufficiently detailed, such that a developer or development team can accomplish the objectives of the request. Once the request is defined and understood, the developer or development team can provide an estimation. Each of the participants indicated that concise, descriptive, and well-written requirements are an effective strategy used to provide accurate estimates to stakeholders. Gaining a complete set of requirements in the initial discussion or evaluation of a request can be difficult. However, the level of detail in a requirement can influence the accuracy of effort for the request estimate. Understanding the requirements mitigates false hope to the requester when the team provides an estimation that may be too low.

Requirements decomposition is a top-down approach used by development teams to identify the objectives of the requested deliverable (McConnell, 2006). Software development professionals decompose requirements into manageable pieces to understand the effort required to complete the task. Development teams use decomposition as a strategy to break the request down into steps or smaller items to provide a more realistic estimation. Developers decompose requirements to gain a granular view of the request. A more granular identification of the activities increases the accuracy of the estimation process through the identification of each step or task in the development request to deliver.

All 10 participants indicated that clear and concise requirements were critical elements to providing accurate estimations. Additionally, 15 of the 20 organization documents reviewed supported the theme of defining and decomposing requirements. Two of the documents I reviewed described a properly written requirement as a request that was complete, unambiguous, consistent, and testable. Eight of the 10 participants indicated that decomposition was a strategy used in providing estimations. After developers decomposed the story and estimated the identified items, they aggregated the values into a final estimation of effort for the request. Table 1 includes frequency information for Theme 1.

Table 1

Frequency of First Major Theme

Major theme	Participant count	Participant references	Document count	Document references
Define and decompose requirements	10	65	15	34

Each of the participants stressed the importance of detailed and complete requirements, and this concept aligned with findings from my literature review. Sehra et al. (2016) stated that requirement uncertainty affected estimation accuracy. Participant 6 indicated that "so whether you are doing the estimation, or you are doing the development, without the full requirements, you really run into a problem." Participant 3 stated that "the biggest problem in providing an estimation was that the requirement was not fully fleshed out" and that "estimation accuracy was dependent on fully defined requirements." Usman et al. (2018) identified unanticipated requirements as one of the four causes of inaccuracies in estimation. Jørgensen (2014) stated that estimation improves though the avoidance of early estimations based on incomplete information. Participant 9 stated that "accurate estimation success was a result of understandable requirements." Participant 9 also indicated that estimation accuracy improved when "the description of the story has success criteria and that the inclusion of testing steps in the story provided developers with an understanding of the functionality in order to better estimate." Providing testing steps gives the developer criteria for acceptance from the perspective of the requester. When prompted to consider a task from the perspective of an outside observer, people are more willing to consider obstacles that they may not otherwise have considered (Wiese et al., 2016).

Information obtained from organization documents supported the theme of define and decompose requirements. One of the organizational project documents stated that the "key activity of the product team is to identify the user request, break requirements down into small valuable stories, and identify clear and concise user acceptance tests and business rules for each story." Wiese et al. (2016) suggested breaking large tasks into smaller subtasks highlights critical steps that may potentially be overlooked. Lévy-Garboua et al. (2018) stated that there is evidence that suggests that as task complexity increases, underestimation becomes more apparent. Decomposition reduces complexity making the estimation process more reliable.

Shmueli and Ronen (2017) noted that both software developers and managers are subject to the planning fallacy. Kahneman and Tversky (1977) indicated that overconfidence increased with a lack of knowledge resulting in an optimistic bias. One of the process and procedures documents reviewed provides instructions on defining the requirements, including the persona of the requests, the specifics of the action to perform, and the expected results. The document supplied instructions to identify minimal items in a requirement and the inclusion of traditional story-based dialog. “As a <type of user>, I want <perform some task>, so that I can <achieve some result>.” Developing a persona in a user request provides the development team with the action, actor, and system(s) affected, enhancing the knowledge the team gains regarding the request, thus providing the information to establish an accurate estimation.

A team role and responsibility document I reviewed stated that before estimation and development, it is necessary to “ensuring the creation of technical user stories in support of the business requirements” and that “acceptance criteria defined upfront drives the development of the software.” The inclusion of success criteria identifies the end state of the request. I reviewed an SDLC document describing the development process as clearly defining the requirements before the estimation stage, “Complete and accurate requirements are desired at this stage and will result in a faster and more efficient development process.” Anooja and Rajawat (2017) suggested that factors such as improved estimation training and higher accuracy of information (requirements) improve estimation. Accurate and complete requirements provide software developers with the information to decompose the request into manageable tasks, thus improving the accuracy of estimates.

My findings support the strategy of developing defined requirements and request decomposition to reduce the phenomena of the planning fallacy. Each of the participants stated that clearly define requirements were necessary for establishing an estimation, and in my literature review, the theme was consistent. Tversky and Kahneman (1974) stated that predictions are often based on an optimistic view of the duration of a previous task, and estimators do not adequately adjust for the demands of a new task that is estimated. Additionally, the findings of a study conducted by Shmueli et al. (2016) provide evidence of manifestations of the planning fallacy in software development projects. Clearly defined requirements reduce the ambiguity of a request and provide the estimator with reliable information to establish an estimation. Decomposition is the process of identifying the different aspects of delivering the solution and breaking them down into manageable items. It is difficult to predict the size of a software project during the initial phases (Shida & Tsuda, 2017) due to incomplete or inaccurate requirements. Sehra et al. (2016) point to inconsistent, incomplete, and unstable requirements as a factor in estimating software development effort. Once the requirements are identified and decomposed, the accuracy of estimates is improved.

The data that I collected and reviewed provided evidence that requirements definition and decomposition are effective strategies to reduce the planning fallacy effects. As stated by Shmueli et al. (2016), decomposition reduced the impact of the planning fallacy. Decomposition is a strategy used by software developers to break a story down into manageable pieces. Eight of the 10 participants I interviewed indicated that an essential strategy in providing estimations was the decomposition of the request. Breaking large tasks into smaller subtasks highlights critical steps that are potentially overlooked otherwise (Wiese et al., 2016). According to (Lévy-Garboua et al., 2018), the greater the complexity, the more difficult the estimation process. Decomposition reduces complexity, thus reducing the difficulty in estimating effort.

According to Tversky and Kahneman (1974), the planning fallacy results from neglecting or ignoring distributional data. Jørgensen (2004) stated that top-down decomposition encourages the outside or distributional history-based thinking. Decomposition is a strategy that allows estimators to identify outside information (distributional) and provide estimations based on reflective assessment. Estimation strategies that use an outside view mitigate the effects of the planning fallacy (Kahneman & Tversky, 1973; Shmueli et al., 2016; Shmueli & Ronen, 2017; Thomas & König, 2018). McConnell (2006) suggests using an approach that decomposes tasks enhances the accuracy and effectiveness of the estimator's judgment. Estimating large tasks is prone to error; thus, decomposition provides higher accuracy. In consideration of the planning fallacy phenomena, incomplete or inaccurate requirements affect the reliability of distributional (outside view) data, thus making estimation potentially unreliable and accuracy problematic.

Theme 2: Reference Historical Data

Referencing historical data was a major theme that was prominent in my study. The use of historical data on previous development requests provides the estimator with quantitative and qualitative information regarding effort on previous similar tasks. Each participant indicated that if relevant historical data were available, they would reference it before providing an estimation. Historical data allows the developer to gauge the complexity of the request. Additionally, using historical information, the developer gains a general idea of how much effort was required previously. Referenced historical data and information on previous work may alert the developer to potential challenges that may arise and act as a reference point to determine complexity. Referencing historical data provides the developer with lessons learned, past experiences, and previous nuances regarding a requested development item. Additionally, referencing historical data can help a developer identify unknowns or potential risks associated with a request. Table 2 includes frequency information for Theme 2.

Table 2

Frequency of Second Major Theme

Major theme	Participant count	Participant references	Document count	Document references
Reference historical data	10	25	6	12

All ten of the participants indicated that referencing historical information was a strategy used in providing estimations. Participant 5 stated, "you do a look back of similar stuff that you've worked on in the past to help give you some identification of what the level of effort is going to be." Participant 5 further added that "based on historical context, is it an easy module, or is it a difficult module." Participant 9 stated regarding the use of historical data that it was beneficial to "to look at past experiences with a similar problem and estimate on that." Participant 6 stated that developers "use previous estimates to estimate the project that we currently have."

Referencing previous information regarding similar work acts as a point of reference to develop an estimation. Participant 3 stated that their strategy in providing estimates was " using our past prior knowledge, we've got a database …so getting that information, plugging it in, and using past experiences." Participant 8 indicated reviewing past similar requests helped the estimator determine complexity. Jørgensen (2004) stated that reviewing other software developers' estimations triggered reflection on the effort the task will require. People make more realistic predictions when they reflect on previous experiences to inform their forecasts.

Five of the participants from three of the organizations indicated that the practice of retrospectives helped establish lessons learned in previous work. The development met team evaluates the iteration in terms of communication, resources, and processes to identify potential areas for improvement (Srivastava & Jain, 2017). All five organizations use a centralized data source or repository to maintain information regarding previous requests and estimations. Some of the tools identified were Jira, Rally, ServiceNow, and SharePoint. Participant 3 stated that “all of our communication is in Jira, so I feel like that's helpful in at least going back and figuring out our estimate” and “keeps our historical context in one place.” The inclusion of historical effort estimation information in future estimations gives greater accuracy in software development estimating (Shmueli et al., 2016). Participant 6 stated the centralized historical information provided the ability to “look back on previous items that are similar so that you can kind of say, Well, A is similar to B, and A took me this amount of effort." Historical data consideration is more likely to bypass a cognitive bias in decision-making (Féris et al., 2017). Participant 9 indicated for estimating, “It does help to have historical data to go by.” Regarding historical information, Participant 6 spoke on the benefits of historical data, “So that allows us to draw a baseline from that experience ... How can we leverage those to estimate this project that we have currently?” In assessing the level of complexity and providing an estimate of effort, development teams benefit from historical information and previous estimation.

The literature I reviewed supported the theme of referencing, capturing, and using historical information. Jørgensen (2014) stated that the accuracy of estimates improves through the use of local context, historical estimation error intervals, and the avoidance of misleading estimation information. Five of the participants use historical data to determine team velocity. Participant 9 indicated that the team used velocity "as sort of a budget" to provide planners with the ability to formulate projected delivery dates. Velocity represents the amount of work that the development team can deliver in a specific time iteration and is a useful predictor of the team's capabilities (Ahmed et al., 2017; Torrecilla-Salinas et al.). Organizations calculate a team's velocity using previous team performance data to determine how much the team can accomplish in each timeframe. Participant 4 stated that" probably the biggest thing is we look at the velocity of our team [for planning purposes]."

Flyvbjerg (2006) introduced the concept of reference class forecasting to improve estimate inaccuracy resulting from bias through considering the actual performance of comparable projects, thereby bypassing the effects of optimistic bias and strategic misrepresentation. Reference class forecasting is an attempt to avoid human bias by relying on historical data from similar past projects as a guideline for predicative estimations. Reference class forecasting is the outside view based on knowledge of the actual performance of referenced comparable projects. Six of the organizational documents that I review supported the theme of referencing historical data. A change management document contained the following importance of capturing historical information: “Each change receives a post-implementation review such as a Lessons Learned or Retrospective. Less than successful changes receive a more extensive review.”

Six organization documents supported the theme of referencing historical data and lessons learned. Team process documents indicate that the "PMO will schedule a brief meeting to discuss lessons learned with the project team." A project management process document stated that the project manager or project lead is required to "Collect and report on metrics on the team's performance over time." According to Srivastava and Jain (2017), areas of potential improvement result from retrospective meetings in which the team evaluates the sprint in terms of communication, resources, and processes. Additionally, a project process document stated that the project manager or team lead would "generate a post-project survey to capture things that went well during the project as well as things that could be improved for a similar project in the future." Information obtained from an SDLC stated that "KPIs provide the organization with trend analysis and identify opportunities for improvement, increased quality, and improved performance." Company-specific calibration and historical data increase accuracy in estimating (Moharreri et al., 2016). Additionally, Jørgensen (2014) stated that the accuracy of estimates improves through the use of local context and the use of historical estimation error intervals.

The theme identified correlates to the conceptual framework of this study, the planning fallacy. The planning fallacy phenomenon occurs when the individual is focusing on the inside view of a task (singular) but not considering the data from an outside perspective of previous tasks (distributional) (Thomas & König, 2018). Participant 4 stated that reflection on past estimations improved their estimation process, “looking back at history and trying to understand how we get closer [in estimates].” Distributional information is primarily a consideration of previous task performance, whereas singular focuses on the task itself (Kahneman & Tversky, 1977; Thomas & König, 2018). Using historical data and prior estimations as a reference are effective strategies used by development teams to mitigate the planning fallacy. According to Kahneman and Tversky (1977), the planning fallacy is the result of underestimation due to neglecting or ignoring distributional, causing an error in prediction. Development teams make decisions based on distributional information to increase the accuracy of the estimate.

Buehler et al. (1994) suggest that people make more realistic predictions when using past experiences to inform their predictions (distributional). The outside view is considering past projects' experience and knowledge to reference similar cases (Shmueli et al., 2016). Each of the participants described the use of previous tasks as an effective strategy in providing estimations. Jørgensen (2004) states that data from past projects, the application of analytics rather than memory, and the use of distributional information (outside view) are strategies to mitigate the planning fallacy. The outside view or reflection on previous experiences is usually more accurate as it bypasses political and cognitive bias (Fridgeirsson, 2016). Effective estimation strategies use historical data to mitigate potential bias and provide software effort estimators with data to give the stakeholders estimations that are more likely to reflect actual effort. The use of historical information (distributional) is an effective strategy used by development teams to mitigate the effects of the planning fallacy.

Theme 3: Identify Potential Risks and Unknowns

Identify risks and unknowns was a theme identified in my study. Risk identification is a standard project management consideration. However, in the context of software development effort estimating, risk can be developers working on new technology, the level of complexity of the module, the number of additional applications the system uses, or the developer's familiarity with the module to modify. Additionally, within the context of Agile software development, teams can begin work with potential unknowns. All 10 participants indicated that they considered risks and unknowns when providing an estimation. Most of the estimation methods used considered risks and unknowns. Five organizational documents I reviewed addressed risk and potential unknowns in software development projects. Table 3 includes frequency information for Theme 3.

Table 3

Frequency of Third Major Theme

Major theme	Participant count	Participant references	Document count	Document references
Identify potential risks and unknowns	10	51	5	15

Participant 3 stated that it was common that "there's some kind of random problem that doesn't work as expected" and that "in software, there are so many unknowns when it comes to any particular project." Participant 8 indicated, "One of the major things, and I am guilty of this, uncertainties lead to a lot of underestimating because we can only estimate what we know or what we foresee." Participant 1 stated that "if it involves cloud or a new technology that we're not familiar with, I have to give some additional time for research." Participant 4 indicated "some sort of subject matter expert or somebody that's got interest or experience in that area." can offset uncertainty.

One of the organizational documents I reviewed included a Roles and Responsibilities guide, which provided information on the duties of a team lead, indicating that they are to “Help identify story dependencies, risks, and possible issues.” Additionally, team leads are to “Collaborate on ideas to address these risks early.” A change management document stated as one of the steps that “Identification of risks to contributing to better estimates of effort, quality of delivery, timeline, and the cost of change.” A software development policy and procedures guide stated that the “Risk evaluation process analysis should be used to determine high-level objectives, risk, cost, and benefits analysis.”

Software development teams use different strategies to address risk in the estimation process. Two participants used T-shirt sizing, 5 participants used story points, 2 participants indicated they used time estimates provided by experts, and 1 participant used a 3-point estimation approach to provide time estimates. Four of the participants noted they additionally used a combination of approaches. Shekhar and Kumar (2016) stated that no single method in software development estimation is considered the best method and suggested using a combination of methods to increase estimation accuracy. Seven of the participants used relative sizing techniques to address risks and unknowns. Relative size measure provides an assessment of complexity rather than effort person-hours.

All five participants who used the story points method used a Fibonacci sequence in the story points approach (1,2,3,5,8,13). The gaps between the sequences provide for a higher degree of uncertainty in the level of effort for larger units of work (Alostad et al., Abdullah, & Aali, 2017; Jadhav et al., 2017; Raslan et al., 2015), the greater the complexity, the higher the level of uncertainty. Essentially, the larger the effort (greater the size), the more likely the error in the estimate; thus, the higher the gap in the sequence (Raslan et al., 2015). Story points are a sizing technique used as a relative unit of measure for expressing the overall size of a user story or development effort. Story points are relative measures rather than quantitative measures (Soni & Kohli, 2017). Two participants in the study used the T-Shirt sizing method of estimation: extra-small, small, medium, large, and extra-large. McConnell (2006) asserted that the t-shirt sizing technique could produce an early estimate to give the business a metric of complexity (size) for determining the level of effort.

Effort-size is a relative measure such as story points, whereas effort-time is an absolute value method, such as person-days or person-hours (Arifin et al., 2017). In story pointing and t-shirt sizing, values indicate complexity and not a measure of time. The relative values can indicate the velocity of a team. Velocity is a measure of how much complexity a team can address over time. Velocity represents the total of story points that the development team can deliver in a specific iteration (Torrecilla-Salinas et al., 2015), and is a useful predictor of the team's capabilities (Ahmed et al., 2017).

The two participants that used the time or person-hours approach indicated that they would pad the estimate based on complexity. Padding is the adding of additional time to the estimate to account for uncertainty. The higher the complexity or more significant the unknown, the larger the padding. The participant that used the 3-point strategy stated that “ You can take the best-case scenario, the worst-case scenario, the most likely with the most weight being applied to the most likely scenario, you can take an average of it.” Usman et al. (2017) suggested providing estimates by averaging three values; fastest, most practical, and maximum values to give a final estimate. Osman and Musa (2016) concurred that different estimation methods are better suited to different development models.

Concerning the planning fallacy, estimate predictions may be optimistic because people do not consider risks and setbacks (Newby-Clark et al., 2000). The participants evaluated uncertainty and unknowns in their methods of calculating estimations. Optimism bias is the belief that there are fewer project risks and an assumption of a more favorable outcome, even in the face of historical information that is contradictory (Pinto, 2013). Underestimation, resulting in optimistic bias, is the lack of consideration of unforeseen circumstances. Optimistic bias can result in underestimation of task effort as unexpected events are not considered or acknowledged. Relative sizing, padding, and three-point estimating are estimation strategies used by developers to offset uncertainty and risk**.**

Theme 4: Communication, Collaboration, and Consensus

Communication, collaboration, and a consensus was a prominent theme identified in my study. Commonly, more than one person is involved in the development of software. Typically, there are many contributors to a software development team in the delivery of a software product. Teams are more productive when they communicate and collaborate in the project. In providing a software development effort estimation, it is a common strategy to use the shared knowledge of the team. All 10 of the interviewed participants indicated that an essential strategy in providing accurate estimations was open communication and team collaboration. Six of the participants stated that it was standard practice for the team to discuss the work item and arrive at a team consensus on the estimation. Additionally, six participants stated that it was a common practice to have a team standup meeting every day as a communication strategy; the remaining four participants indicated that the team meets at least two times a week for a status reporting and discussion on the project under development.

Nine organizational documents included information regarding the practice of effective and frequent communication as a standard event in software development. Eight of the organizations used a change management application such as Jira, Rally, or ServiceNow to provide timely visual indicators of project status and progress and capture requirements, estimations, and team communication. Additionally, eight of the participants used online communication tools for messaging and video conferencing.

These tools included Slack, Zoom, and Microsoft Teams in addition to e-mail, and face to face discussions and team meetings. All 10 of the participants indicated that frequent communication with product owners and project managers was an essential strategy in maintaining schedules and commitments. Table 4 includes frequency information for Theme 4.

Table 4

Frequency of Four Major Theme

Major theme	Participant count	Participant references	Document count	Document references
Communication, collaboration and consensus	10	52	9	25

Teams collaborate and communicate to share information. Participant 6 stated that "having an established routine in meeting with the team and discussing status actually does benefit the team." Participant 6 went on to add that collaboration "help[ed] improve the estimations as well because you're able to get a little feedback here and there from your teammates and then figure out maybe something you didn't think of" was a common practice. Participant 4 indicated that the organization was supportive of "promoting good communication between team members." Additionally, participant 4 further stated that:

> My team gets together right before our sprints, and we really go through and look at what those tasks are. We all get together, and we look at what tasks we've got coming up and get input from those team members as to how much time we think that's going to take".

Communication, collaboration, and consensus are essential strategies used by teams to provide estimations, share knowledge, and gain valuable feedback.

Five of the participants stated that they used a planning poker approach in the estimation process. According to Taylor (2016), the estimation method uses a consensus approach to estimate development effort that minimizes peer pressure. Planning poker can consist of several rounds of discussion and re-estimation to reach consensus (Bilgaiyan et al., 2017; Choetkiertikul et al., 2018). Participant 9 described their estimation process:

> The development lead or project manager presents the story, the requirements, and then we play Agile Poker. Every one of us provides a story point estimation on the requirement as presented. The person that estimates the lowest gives a reason why they estimated that as low as they did. Then the person that estimated the highest gives a reason why they estimated as high, and then a determination is agreed upon on what the levels should be.

Participant 5 indicated that the team participated in online meetings to estimate stories:

> 'I will share the ticket, and we'll all review in Jira ... we've used a variety of planning poker tools, and to be honest with you, I find that the screen sharing with Jira to be the most effective. There are a few tools out there, but I generally prefer just sharing the ticket and reading it over, allow the developers to ask questions about it. Once those questions are answered, have everybody throw out a number either via Slack or type it into a Google sheet and then discuss why those

> estimates are different and that the team will discuss and hash it out until we come to some sort of agreement.

Participant 3 stated that during a group-based estimation session, "At that time, we have all developers in the team, anyone who's contributing to the project join the call and they chime in." Additionally, participant 3 went on to explain, "I think the task is going to be a medium task. So, anyone can challenge that to say why do you consider that as a medium, why not a smaller, why not a large." Jørgensen (2014) stated that the accuracy of estimates improves the conducting of a group-based approach. Participant 1 indicated that communication and collaboration with the business analyst was a standard practice "I think it all starts with getting involved early in the projects with the requirements, even in the requirements gathering. When a project starts, we're in constant communication with the business analyst, sharing dialogue back and forth." Agile traditionally incorporates extensive user involvement in the development process (Taylor, 2016). Participant 1 went on to further state "The team uses these sessions to discuss the requirements, potential solutions or approaches, and any questions or feedback for the user." The agile approach to software development consists of self-organized teams focusing on collaboration and communication (Vallon et al., 2018).

An SDLC document I reviewed stated that communication with the stakeholders provided a strategy to define the work requested. The SDLC document contained the following statement:

> "Interview the business stakeholders to determine what business problem is to be addressed and to translate and document the stakeholder requirements and

> preferences into language the technical development team can use to build a list of the specific features and deliverables of the solution."

A software development procedure guide I reviewed indicated that the team lead or scrum master facilitated the communication protocols to ensure effectiveness. An organizational roles and responsibilities document instructed team leads to "Create an atmosphere that is collaborative, fun, challenging yet rewarding." Additionally, the document further stated that the team leader "Collaborates with the team and business users to ask questions, get clarification, provide input, share progress, provide timeline commitments and bring visibility to any daily impediments or issues that are standing in the way of performance." One of the Agile Manifesto tenets is that collaboration with customers is more important than contract negotiation (Coleman, 2016; Drury-Grogan et al., 2017).

According to (Usman et al., 2017), a factor in software development estimation is the level of communication with the customer. All participants indicated that either direct communication or team leadership communication with the customer was standard practice. Jørgensen (2014) stated that the accuracy of estimates improves using a group-based approach. Eight of the participants said that their team used group-based estimation strategies in their estimation process. Prakash and Viswanathan (2017) indicated that characteristics of successful agile estimating include collaboration with product owners, estimations accomplished by a team rather than an individual.

Kahneman and Tversky (1977) indicated that judgments should be driven from a reflective assessment rather than from immediate impressions, although intuition from a knowledgeable professional is beneficial. Teams use communication and collaboration to gain insight from other developers and provide a perspective beyond personal opinions. The planning fallacy phenomenon occurs when the individual is focusing on the inside view of a task (singular) but not considering the data from an outside perspective of previous tasks (distributional) (Thomas & König, 2018). A collaborative approach provides distribution knowledge that a single individual may not possess or consider. (Jørgensen, 2004). Team-based estimation sessions provide a forum for software teams to collectively evaluate the complexity level to arrive at a consensus on the estimation.

Optimistic bias is more prevalent in the estimation of one's effort. Many studies on human judgment prove that people are generally over-optimistic in predicting their performance (Jørgensen, 2004). People have a propensity to underestimate their effort but not others' effort (Buehler et al., 1994). Group based estimation strategies reduce optimistic bias as the estimation includes assessments from other developers and not decided singularly. Communication, collaboration, and consensus are strategies identified by the participants to provide accurate estimates by reducing the effects of the planning fallacy and personal optimistic bias.

Applications to Professional Practice

The specific IT problem that was the bases of this study is the lack of effective strategies used by software development professionals in providing accurate effort estimations for software development. Participant interviews and organizational documents provided data for analysis to uncover effective strategies used in the estimation process. There are a variety of different methods that teams use in determining an estimate. However, the data resulting from my study suggests that there are common strategies used by software development professionals in determining accurate effort estimates. Although effort estimations are not exact, the results of this study conclude that there are common elements used by software development professionals in assessing the level of effort of a software development request. Using the information from the interviews and organizational documents, I identified four primary themes: define and decompose requirements, reference historical data, identify risks and unknowns, and communication collaborations, and consensus. As discussed by the participants and identified in organizational documents, these themes were essential elements in the strategies used and, when used in combination, allowed the estimators to provide estimations that aligned with actual effort values.

IT organizations engaged in software development can use the results of this study to train and coach software teams in practical strategies to increase the accuracy of estimates that they provide. Organizations can encourage a culture that supports the developers with tools and processes that promote consideration of the strategies discussed. Software development estimation is an art rather than an exact science. Thus, strategies identified in this study can be supported by organizations to establish foundational elements in their estimation process. Additionally, the strategies discussed can serve as points of consideration when providing estimations. All four of the strategies identified and discussed in this study point to the need to consider distributional information when providing estimates. The use of distributional data, in addition to a singular view of the task, gives the developer the information to make a more accurate assessment of the level of effort a request, thus providing a more reliable estimate.

Organizations can use the findings from this study in establishing policies and procedures for their software development teams. The themes identified in this study could be adopted by organizations to support developers in the estimation process. The use of historical data, considerations of risk and unknowns, effective collaboration and communication, and that decomposition of the requirements are effective strategies in effort estimating. Organizations can support using these strategies by providing the development teams with the tools and training using the information gained in this study. This study's conclusions may better equip the estimators to provide project managers and stakeholders with more accurate estimates and more realistic timelines to predict a delivery schedule. Additionally, project managers have a more realistic timeline that is likely to be more reliable when answering the questions of “when will this be done.”

Developers can use the strategies identified in this study to provide estimates that more closely align with actual time spent in the completion of a software development request. Software development professionals can use the strategies identified in this study to gain a more in-depth consideration of distributional information and its criticality in the estimation process. In the use of historical data, developers are more likely to provide more realistic effort estimations. Thus, project managers can have greater confidence in the accuracy of the estimates provided. Estimates that are more accurate increase product quality, customer satisfaction, work-life balance for the team, and give organizations a more realistic view of budgets, cost, and delivery planning.

Implications for Social Change

Before beginning my data collection, my initial expectations for social change were that using effective estimation strategies would reduce the stress level of IT managers, project stakeholders, and software development professionals. Additionally, estimates that are more reflective of actual effort can benefit development teams by improving morale, work-life balance, alignment of expectations, and software quality. Beyond what I initially anticipated, I also found that development teams gained a higher level of personal satisfaction when they meet commitments and that discussing estimations gave teams a reason to collaborate more. Team collaboration promotes higher team satisfaction, engagement, and trust. The participants I spoke with seemed less anxious about providing estimations than I expected and had higher confidence in the estimates that they offered.

Using the strategies discussed in this study reduced stress in the delivery process and gave the project managers a higher level of confidence in meeting organizational goals. Moral among the developers was high, and each described a sense of community and engagement with the other team members and organizational leadership. I gathered from the discussions that the project managers had a high level of trust in the estimates provided, thus gaining confidence in providing the stakeholders with realistic expectations of delivery. Additionally, increased communication and collaboration result in less stress to the developers, organization, and customers. I believe that estimations that are inaccurate lead to late projects. Late projects lead to unhappy customers, overworked staff, and reduce the quality of the delivered product. Using effective strategies in effort estimating, developers and managers gain a higher level of confidence in the planning, cost forecasting, and an increase in the quality of software development changes giving customers and stakeholders a higher level of satisfaction. Accurate estimations provide the foundation of meeting product objectives and commitments.

Positive social change can result from this study to improve the lives of those engaged in software estimation using effective strategies that produce accurate effort estimations for software development delivery. The implications for positive social change are that increased accuracy of software effort estimation could reduce the stress level of IT managers, project stakeholders, and software development professionals by providing more realistic timeframes for software delivery. The study may provide positive social changes, as estimates that are more reflective of actual effort can benefit development teams by improving morale, work-life balance, alignment of expectations, and software quality. The contributions made through this research may provide development managers and development teams with practical and effective strategies for accurate effort estimations.

Recommendations for Action

Software development professionals, team leads, and project managers should review and consider how their development teams incorporate the strategies identified in this study within their estimation process. Each of the identified themes provides building blocks to practical strategies in estimation. Underestimation is more problematic in software development as developers may provide estimates before having sufficient knowledge of the requested change. Additionally, the lack of historical information, an evaluation of effort based solely on singular information, and the absence of risk analysis have adverse effects on the accuracy of estimates. The recommendations obtained from this study offer effective strategies to mitigate the impact of the planning fallacy and offset the causes of potential optimistic bias. The data from this study identify strategies used by software development teams in the estimation process and offer evidence that distributional data has a positive effect on the accuracy of estimates.

Organizational leadership can adopt the suggestions of the study within their process and procedures documentation for effort estimation. Additionally, the organizations can provide training to estimators to ensure they are aware of the need for distributional data, collaboration, and understanding of risks in the estimation process. Additionally, organizational software development process and procedures documents can instruct practitioners in the value of decomposition in providing estimations. Organizations can structure their estimation practices to use a team-based collaborative approach in which multiple professionals have input. The team members discuss the estimates, and the teams reach a consensus on the final estimation.

Recommendations for Further Study

There are several recommendations for future research derived from the limitations indicated in this research. Additionally, I propose recommendations that arose from the findings of this research. This study was limited to ten participants in five organizations within the region of South Texas and may not be generalizable to all software development teams. The study was limited to small- to medium-sized organizations. The following are my recommendations for future research. The finding of this study warrant exploration of strategies used by large corporations. Second, I recommend expanding the research to other geographical areas. Third, I would suggest performing the analysis using a larger sample size. Additionally, expanding the study to include organizations that do not have an effective strategy to determine if the themes identified exist or are absent.

Future research may include quantitative data to evaluate levels of under or overestimation. Additionally, further research may consider the size of the team on the effectiveness of estimates. As well, future research may address the involvement of a product owner or business analyst in the accuracy of estimations. Additional research may investigate what type of methods organizations use to calculate or determine the level of effort in their estimation process. All the participants in my study were male. Although I did not feel having an equal gender distribution would affect my results, additional research may provide evidence to the contrary.

Reflections

As a professional who has worked as a software engineer, software development manager, and as a project manager, I understand the need to provide estimations and that the estimates are as accurate as possible. I also know that delivering accurate estimations can be difficult, especially within the context of an Agile environment. Being involved in the process of estimating as well as the receiver of estimations, I understand the challenges. Estimates are not commitments, but rather, an approximation.

In accomplishing this study, I was surprised to discover that in my discussing the subject of estimation, many organizations I spoke with to determine eligibility for the research indicated that their approach was unsatisfactory. Some even stated that they had decided to abandon estimations due to the inaccuracy of estimates provided. I found this somewhat surprising as, without some type of estimate, planning, budgeting, and scheduling would be very problematic.

All the participants in my study provide knowledge regarding effective strategies that they use in the estimation process. I felt that the semi-structured interview approach was practical as it allowed the participants to describe the effective estimation strategies they use. The interview questions and approach reduced the potential bias I made have had in the process of estimating. Before the interviews, I did think that the focus of the discussion would be on a specific method used by the participants. However, as I conducted my interviews, I began to understand that although practices and effort estimation methods may differ, the strategies used were common.

Summary and Study Conclusions

Estimating effort within the context of Agile software development is more problematic than traditional waterfall software development projects. However, the findings of this research point to common strategies that developers use to provide project managers and project planners with more accurate estimates. The strategies that have a positive effect on estimation are first, software requests should be detailed and decomposed such that the items identified to accomplish the task are adequately defined and broken down into manageable tasks. Accurately detailed requirements are essential in decomposition, and decomposition provides an effective strategy for accurate estimating. Second, when determining an estimation, identifying the risk and unknowns involved in completing the request should be considered in the estimation process. Potential risks and unknowns can require additional effort not accounted for in an estimation, thus adversely affecting the accuracy of the estimate provided. Third, capturing data and information about the development request and making the data available enhances future estimations' accuracy. Using lessons learned and retrospectives provide needed feedback to improve the estimation process. Referencing previous information can highlight information to consider and evaluate before giving an estimate to enhance the accuracy of the estimation provided. Finally, communication and collaboration within the team in assessing the effort required to accomplish the task increases the accuracy in the evaluation of the effort. Involving the entire team in providing an estimate reduces the single mindset or bias of one individual and allows for the consideration of multiple viewpoints.

Embracing, supporting, and establishing the strategies identified in this study provides a development team with more distributional data, outside views, opinions, and perspectives that may otherwise be overlooked or not considered. The information identified in this study provides strategies to mitigate the effects of the planning fallacy. Although estimation methods differ among organizations, there is a commonality in the strategies identified in this study to provide accurate estimations. The use and consideration of these strategies will likely benefit organizations, personnel, product quality, and quality of life for all those involved.

References

Aagaard, J., & Matthiesen, N. (2015). Methods of materiality: Participant observation and qualitative research in psychology. *Qualitative Research in Psychology*, *13*(1), 33–46. https://doi.org/10.1080/14780887.2015.1090510

Abdalhamid, S., & Mishra, A. (2017). Adopting of agile methods in software development organizations: Systematic mapping. *TEM Journal*, *6*(4), 817–825. https://doi.org/10.18421/TEM64-22

Abdalla, M. M., Oliveira, L. G. L., Azevedo, C. E. F., & Gonzalez, R. K. (2018). Quality in qualitative organizational research: Types of triangulation as a methodological alternative. *Administração: Ensino e Pesquisa*, *19*(1), 66–98. https://doi.org/10.13058/raep.2018.v19n1.578

Abdullah, A. A., & Qureshi, R. (2018). The proposed L-Scrumban methodology to improve the efficiency of agile software development. *International Journal of Information Engineering and Electronic Business*, *10*(3), 23–35. https://doi.org/10.5815/ijieeb.2018.03.04

Abualkishik, A. Z., Ferrucci, F., Gravino, C., Lavazza, L., Liu, G., Meli, R., & Robiolo, G. (2017). A study on the statistical convertibility of IFPUG Function Point, COSMIC Function Point and Simple Function Point. *Information and Software Technology*, *86*, 1–19. https://doi.org/10.1016/j.infsof.2017.02.005

Abualkishik, A. Z., & Lavazza, L. (2018). IFPUG function points to COSMIC function points convertibility: A fine-grained statistical approach. *Information and Software Technology*, *97*(July 2017), 179–191. https://doi.org/10.1016/j.infsof.2018.01.012

Adnan, M., & Afzal, M. (2017). Ontology based multiagent effort estimation system for scrum agile method. *IEEE Access, 5,* 25993–26005. https://doi.org/10.1109/ACCESS.2017.2771257

Ahmad, M. O., Dennehy, D., Conboy, K., & Oivo, M. (2018). Kanban in software engineering: A systematic mapping study. *Journal of Systems and Software*, *137*, 96–113. https://doi.org/10.1016/j.jss.2017.11.045

Ahmed, A. R., Tayyab, M., Bhatti, S. N., Alzahrani, A. J., & Babar, M. I. (2017). Impact of story point estimation on product using metrics in scrum development process. *International Journal of Advanced Computer Science and Applications*, *8*(4), 385–391. https://doi.org/10.14569/IJACSA.2017.080452

Alase, A. (2017). The interpretative phenomenological analysis (IPA): A guide to a good qualitative research approach. *International Journal of Education and Literacy Studies*, *5*(2), 9-19. https://doi.org/10.7575/aiac.ijels.v.5n.2p.9

Allen, R. E. S., & Wiles, J. L. (2016). A rose by any other name: Participants choosing research pseudonyms. *Qualitative Research in Psychology*, *13*(2), 149–165. https://doi.org/10.1080/14780887.2015.1133746

Almakadmeh, K., Al-Sarayreh, K. T., & Meridji, K. (2018). A measurement model of the functional size of software maintainability requirements. *Journal of Theoretical and Applied Information Technology*, *96*(12), 3829–3845. Retrieved from https://www.jatit.org/

Alostad, J. M., Abdullah, L. R. A., & Aali, L. S. (2017). A Fuzzy based model for effort estimation in scrum projects. *International Journal of Advanced Computer Science* *and Applications*, *8*(9), 270–277. Retrieved from www.ijacsa.thesai.org

Alqudah, M., & Razali, R. (2017). Key factors for selecting an Agile method: A systematic literature review. *International Journal on Advanced Science, Engineering and Information Technology*, *7*(2), 526-537. https://doi.org/10.18517/ijaseit.7.2.1830

Al-Zewairi, M., Biltawi, M., Etaiwi, W., & Shaout, A. (2017). Agile software development methodologies: Survey of surveys. *Journal of Computer and Communications*, *05*(05), 74–97. https://doi.org/10.4236/jcc.2017.55007

Andersen, B., Samset, K., & Welde, M. (2016). Low estimates – high stakes: Underestimation of costs at the front-end of projects. *International Journal of Managing Projects in Business*, *9*(1), 171–193. https://doi.org/10.1108/IJMPB-01-2015-0008

Anheier, H. (2016). Of hiding hands and other ways of coping with uncertainty: A commentary. *Social Research*, *83*(4), 1005–1011. Retrieved from https://www.muse.jhu.edu/article/649543

Anooja, A., & Rajawat, S. (2017). Comparative analysis of software cost-effort estimation and agile in perspective of software development. *International Journal of Advanced Research in Computer Science*, *8*(8), 121–125. https://doi.org/10.26483/ijarcs.v818.4666

Anwer, F., & Aftab, S. (2017). Latest customizations of XP: A systematic literature review. *International Journal of Modern Education and Computer Science*, *9*(12), 26–37. https://doi.org/10.5815/ijmecs.2017.12.04

Arifin, H. H., Daengdej, J., & Khanh, N. T. (2017). An empirical study of effort-size and effort-time in expert-based estimations. *Proceedings - 8th IEEE International Workshop on Empirical Software Engineering in Practice, IWESEP 2017*, 35–40. https://doi.org/10.1109/IWESEP.2017.21

Arsel, Z. (2017). Asking questions with reflexive focus: A tutorial on designing and conducting interviews. *Journal of Consumer Research*, *44*(4), 939–948. https://doi.org/10.1093/jcr/ucx096

Awasthy, R. (2015). Journey of doing quasi-ethnographic study in organizations Beginning of My PhD Journey. *Vision*, *19*(3). https://doi.org/10.1177/0972262915593667

Azanha, A., Argoud, A. R. T. T., de Camargo Jr, J. B., & Antoniolli, P. D. (2017). Agile project management with Scrum: A case study of a Brazilian pharmaceutical company IT project. *International Journal of Managing Projects in Business*, *10*(1), 121–142. https://doi.org/10.1108/IJMPB-06-2016-0054

Azzeh, M., & Nassif, A. B. (2016). A hybrid model for estimating software project effort from use case points. *Applied Soft Computing Journal*, *49*, 981–989. https://doi.org/10.1016/j.asoc.2016.05.008

Babchuk, W. A. (2019). Fundamentals of qualitative analysis in family medicine. *Family Medicine and Community Health, 7(2)*, https://doi.org/10.1136/fmch-2018-000040

Barrett, D., & Twycross, A. (2018). Data collection in qualitative research. *Evidence Based Nursing*, *21*(3), 63–64. https://doi.org/10.1136/eb-2018-102939

Baruah, N. (2015). Requirement management in agile software environment. *Procedia*

Computer Science, *62*(Scse), 81–83. https://doi.org/10.1016/j.procs.2015.08.414

Baseer, K. K., Reddy, A. R. M., & Bindu, C. S. (2015). A systematic survey on waterfall vs. agile vs. lean. *I-Manager's Journal on Software Engineering*, *9*(3), 34–59. Retrieved from http://www.imanagerpublications.com/JournalIntroduction.aspx?journal=JournalonSoftwareEngineering

Bilgaiyan, S., Mishra, S., & Das, M. (2016). A review of software cost estimation in agile software development using soft computing techniques. *2016 2nd International Conference on Computational Intelligence and Networks (CINE)*, 112–117. https://doi.org/10.1109/CINE.2016.27

Bilgaiyan, S., Sagnika, S., Mishra, S., & Das, M. (2017). A systematic review on software cost estimation in agile software development. *Journal of Engineering Science and Technology Review*, *10*(4), 51–64. https://doi.org/10.25103/jestr.104.08

Blalock, A. E. (2018). Incorporating Critical Qualitative Inquiry in Nonprofit Management Education. *Administrative Theory and Praxis*, *40*(1), 43–59. https://doi.org/10.1080/10841806.2017.1420839

Boby, J. B., Kadadevaramath, R. S., & Edinbarough, I. A. (2017). Designing software development processes to optimize multiple output performance characteristics. *Software Quality Professional*, *19*(4), 16–24. Retrieved from http://www.asq.org

Boddy, C. R. (2016). Sample size for qualitative research. *Qualitative Market Research*, *19*(4), 426–432. https://doi.org/10.1108/QMR-06-2016-0053

Boehm, B., Abts, C., Brown, A. W., Chulani, S., Clark, B., Hoirowitz, E., … Steece, B.

(2000). *Software cost estimation with Cocomo II*. Upper Saddle River, NJ: Prentice Hall PTR.

Bracken-Roche, D., Bell, E., Macdonald, M. E., & Racine, E. (2017). The concept of “vulnerability” in research ethics: An in-depth analysis of policies and guidelines. *Health Research Policy and Systems*, *15*(1), 1–18. https://doi.org/10.1186/s12961-016-0164-6

Brad, M. C., Birloi, F., Bratulescu, A., & Blaga, I. B. (2016). A comparative study of agile project management software tools. *Economy Informatics*, *16*(1), 27–39. Retrieved from http://www.economyinformatics.ase.ro

Brodsky, A., & Amabile, T. M. (2018). The downside of downtime: The prevalence and work pacing consequences of idle time at work. *Journal of Applied Psychology*, *103*(5), 496–512. https://doi.org/10.1037/apl0000294

Brooks, J. S., & Normore, A. H. (2015). Qualitative research and educational leadership: Essential dynamics to consider when designing and conducting studies. *International Journal of Educational Management*, *29*(7), 798–806. https://doi.org/10.1108/IJEM-06-2015-0083

Brown, A., & Danaher, P. A. (2019). CHE Principles: facilitating authentic and dialogical semi-structured interviews in educational research. *International Journal of Research and Method in Education*, *42*(1), 76–90. https://doi.org/10.1080/1743727X.2017.1379987

Browne, G. J., Appan, R., Safi, R., & Mellarkod, V. (2018). Investigating illusions of agreement in group requirements determination. *Information and Management*, *55*(September 2016), 1071–1083. https://doi.org/10.1016/j.im.2018.05.013

Bruno, A., & Dell'Aversana, G. (2017). Reflective practice for psychology students: The use of reflective journal feedback in higher education. *Psychology Learning and Teaching*, *16*(2), 248–260. https://doi.org/10.1177/1475725716686288

Buehler, R., Griffin, D., Lam, K. C. H., & Deslauriers, J. (2012). Perspectives on prediction: Does third-person imagery improve task completion estimates? *Organizational Behavior and Human Decision Processes*, *117*(1), 138–149. https://doi.org/10.1016/j.obhdp.2011.09.001

Buehler, R., Griffin, D., & MacDonald, H. (1997). The role of motivated reasoning in optimistic time predictions. *Personality and Social Psychology Bulletin*, *23*(3), 238–247. https://doi.org/10.1177/0146167297233003

Buehler, R., Griffin, D., & Ross, M. (1994). Exploring the "planning fallacy": Why people underestimate their task completion times. *Journal of Personality and Social Psychology*, *67*(3), 366–381. https://doi.org/10.1037/0022-3514.67.3.366

Buehler, R., Messervey, D., & Griffin, D. (2005). Collaborative planning and prediction: Does group discussion affect optimistic biases in time estimation? *Organizational Behavior and Human Decision Processes*, *97*(1), 47–63. https://doi.org/10.1016/j.obhdp.2005.02.004

Buehler, R., Peetz, J., & Griffin, D. (2010). Finishing on time: When do predictions influence completion times? *Organizational Behavior and Human Decision Processes*, *111*(1), 23–32. https://doi.org/10.1016/j.obhdp.2009.08.001

Bureau of Labor Statistics - Software Developers, Application. (2019). Retrieved June

25, 2019, from https://www.bls.gov/oes/current/oes151132.htm#st

Busse, C., Kach, A. P., & Wagner, S. M. (2017). Boundary conditions: What they are, how to explore them, why we need them, and when to consider them. *Organizational Research Methods*, *20*(4), 574–609. https://doi.org/10.1177/1094428116641191

Butt, S. A. (2016). Study of agile methodology with the cloud. *Pacific Science Review B: Humanities and Social Sciences*, *2*(1), 22–28. https://doi.org/10.1016/j.psrb.2016.09.007

Candela, A. G. (2019). Exploring the function of member checking. *The Qualitative Report*, *24*(3), 3–24. Retrieved from https://nsuworks.nova.edu/tqr/vol24/iss3/14

Carminati, L. (2018). Generalizability in qualitative research: A tale of two traditions. *Qualitative Health Research*, *28*(13), 2094–2101. https://doi.org/10.1177/1049732318788379

Chen, K., & Rea, A. (2018). Do pair programming approaches transcend coding? Measuring agile attitudes in diverse information systems courses. *Journal of Information Systems Education*, *29*(2), 53–65. Retrieved from http://jise.org

Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T. T. M., Ghose, A., & Menzies, T. (2018). A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, *14*(8). https://doi.org/10.1109/TSE.2018.2792473

Chung, E. Y. (2019). Identifying evidence to define community-based rehabilitation practice in China using a case study approach with multiple embedded case study design. *BMC Health Services Research*, *19*(1), 1–11.

https://doi.org/10.1186/s12913-018-3838-7

Coleman, G. (2016). Agile software development. *Software Quality Professional*, *19*(1), 23–29. Retrieved from http://asq.org

Collins, C. S., & Stockton, C. M. (2018). The central role of theory in qualitative research. *International Journal of Qualitative Methods*, *17*(1), 1–10. https://doi.org/10.1177/1609406918797475

Creswell, J. W., & Poth, C. N. (2018). *Qualitative inquiry & research design: Choosing among five approaches* (4th ed.). Thousand Oaks, CA: Sage Publications, Inc.

Cunha, J. A. O. G., Moura, H. P., & Vasconcellos, F. J. S. (2016). Decision-making in Software Project Management: A Systematic Literature Review. *Procedia Computer Science*, *100*, 947–954. https://doi.org/10.1016/j.procs.2016.09.255

Cypress, B. S. (2017). Rigor or reliability and validity in qualitative research: Perspectives, strategies, reconceptualization, and recommendations. *Dimensions of Critical Care Nursing*, *36*(4), 253–263. https://doi.org/10.1097/DCC.0000000000000253

Cypress, B. S. (2019). Data analysis software in qualitative research. *Dimensions of Critical Care Nursing*, *38*(4), 213–220. https://doi.org/10.1097/DCC.0000000000000363

Dasgupta, M. (2015). Exploring the relevance of case study research. *Vision*, *19*(2). https://doi.org/10.1177/0972262915575661

Dennehy, D., & Conboy, K. (2017). Going with the flow: An activity theory analysis of flow techniques in software development. *Journal of Systems and Software*, *133*, 160–173. https://doi.org/10.1016/j.jss.2016.10.003

Denscombe, M. (2013). The role of research proposals in business and management education. *International Journal of Management Education*, *11*(3), 142–149. https://doi.org/10.1016/j.ijme.2013.03.001

Deterding, N. M., & Waters, M. C. (2018). Flexible coding of in-depth interviews: A twenty-first-century approach. *Sociological Methods and Research*. https://doi.org/10.1177/0049124118799377

Dewi, R. S., & Subriadi, A. P., (2017a). A modification complexity factor in function points method for software cost estimation towards public service application. *Procedia Computer Science*, *124*, 415–422. https://doi.org/10.1016/j.procs.2017.12.172

Dewi, R. S., & Subriadi, A. P. (2017b). A comparative study of software development size estimation method: UCPabc vs function points. *Procedia Computer Science*, *124*, 470–477. https://doi.org/10.1016/j.procs.2017.12.179

Di Martino, S., Ferrucci, F., Gravino, C., & Sarro, F. (2016). Web Effort Estimation: Function Point Analysis vs. COSMIC. *Information and Software Technology*, *72*, 90–109. https://doi.org/10.1016/j.infsof.2015.12.001

Dönmez, D., & Grote, G. (2018). Two sides of the same coin – how agile software development teams approach uncertainty as threats and opportunities. *Information and Software Technology*, *93*, 94–111. https://doi.org/10.1016/j.infsof.2017.08.015

dos Santos, P. S. M., Beltrão, A. C., de Souza, B. P., & Travassos, G. H. (2018). On the benefits and challenges of using kanban in software engineering: a structured synthesis study. *Journal of Software Engineering Research and Development*, *6*(1), 1–29. https://doi.org/10.1186/s40411-018-0057-1

Dragicevic, S., Celar, S., & Turic, M. (2017). Bayesian network model for task effort estimation in agile software development. *Journal of Systems and Software*, *127*, 109–119. https://doi.org/10.1016/j.jss.2017.01.027

Drury-Grogan, M. L., Conboy, K., & Acton, T. (2017). Examining decision characteristics & challenges for agile software development. *Journal of Systems and Software*, *131*, 248–265. https://doi.org/10.1016/j.jss.2017.06.003

Ebert, C., & Paasivaara, M. (2017). Scaling agile. *IEEE Software*, *34*(6), 98–103. https://doi.org/10.1109/MS.2017.4121226

Ellis, P. (2016). The language of research (part 8): Phenomenological research. *Wounds UK*, *12*(1), 128–129. Retrieved from https://www.wounds-uk.com/

Etikan, I., Musa, S. A., & Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American Journal of Theoretical and Applied Statistics*, *5*(1), 30–44. https://doi.org/10.11648/j.ajtas.20160501.11

Farah-Stapleton, M., Auguston, M., & Giammarco, K. (2016). Executable behavioral modeling of system and software architecture specifications to inform resourcing decisions. *Procedia Computer Science*, *95*, 48–57. https://doi.org/10.1016/j.procs.2016.09.292

Fayard, A., & Van Maanen, J. (2015). Making culture visible: reflections on corporate ethnography. *Journal of Organizational Ethnography*, *4*(1), 4–27. https://doi.org/10.1108/JOE-12-2014-0040

Féris, M. A. A., Zwikael, O., & Gregor, S. (2017). QPLAN: Decision support for evaluating planning quality in software development projects. *Decision Support Systems*, *96*, 92–102. https://doi.org/10.1016/j.dss.2017.02.008

Flannery, M. (2016). Common perspectives in qualitative research. *Oncology Nursing Forum*, *43*(4), 517–518. https://doi.org/10.1188/16.ONF.517-518

Flyvbjerg, B. (2006). From Nobel Prize to project management: getting risks right. *Project Management Journal*, *34*(3), 5–15. https://doi.org/10.1177/875697280603700302

Flyvbjerg, B. (2013). Quality control and due diligence in project management: Getting decisions right by taking the outside view. *International Journal of Project Management*, *31*(5), 760–774. https://doi.org/10.1016/j.ijproman.2012.10.007

Flyvbjerg, B. (2016). The fallacy of beneficial ignorance: A test of Hirschman's hiding hand. *World Development*, *84*, 176–189. https://doi.org/10.1016/j.worlddev.2016.03.012

Flyvbjerg, B. (2018). Planning fallacy or hiding hand: Which is the better explanation? *World Development*, *103*, 383–386. https://doi.org/10.1016/j.worlddev.2017.10.002

Flyvbjerg, B., Ansar, A., Budzier, A., Buhl, S., Cantarelli, C., Garbuio, M., … van Wee, B. (2018). Five things you should know about cost overrun. *Transportation Research Part A: Policy and Practice*. https://doi.org/10.1016/j.tra.2018.07.013

Flyvbjerg, B., & Sunstein, C. R. (2015). The principle of the malevolent hiding hand; or, the planning fallacy writ large. *Social Research*, *83*(4). https://doi.org/10.2139/ssrn.2654217

Fox, R. (2016). The true cost. *Digital Library Perspectives*, *32*(4), 232–238. https://doi.org/10.1108/DLP-07-2016-0018

Francis-Smythe, J. A., & Robertson, I. T. (1999). On the relationship between time management and time estimation. *British Journal of Psychology*, *90*(3), 333–347. https://doi.org/10.1348/000712699161459

Freire, A., Perkusich, M., Saraiva, R., Almeida, H., & Perkusich, A. (2018). A Bayesian networks-based approach to assess and improve the teamwork quality of agile teams. *Information and Software Technology*, *100*(March), 119–132. https://doi.org/10.1016/j.infsof.2018.04.004

Fridgeirsson, T. V. (2016). Reference class forecasting in Icelandic transport infrastructure projects. *Transport Problems*, *11*(2), 103–115. https://doi.org/10.20858/tp.2016.11.2.10

Fusch, P., Fusch, G. E., & Ness, L. R. (2018). Denzin 's paradigm shift: Revisiting triangulation in qualitative research. *Journal of Social Science*, *10*(1), 19–32. https://doi.org/10.5590/JOSC.2018.10.1.02

Fustik, V. (2017). The advantages of agile methodologies applied in the ICT development projects. *International Journal on Information Technologies & Security*, *9*, 51–63. Retrieved from http://ijits-bg.com/

Green, C. A., Duan, N., Gibbons, R. D., Hoagwood, K. E., Palinkas, L. A., & Wisdom, J. P. (2015). Approaches to Mixed Methods Dissemination and Implementation Research: Methods, Strengths, Caveats, and Opportunities. *Administration and Policy in Mental Health and Mental Health Services Research*, *42*(5), 508–523.

https://doi.org/10.1007/s10488-014-0552-6

Guetterman, T. C., & Fetters, M. D. (2018). Two methodological approaches to the integration of mixed methods and case study designs: A systematic review. *American Behavioral Scientist*, *62*(7), 900–918. https://doi.org/10.1177/0002764218772641

Haines, T., Idenudia, E., & Raisinghani, M. S. (2017). The conceptual model for agile tools and techniques. *American Journal of Management*, *17*(2007), 77–88. Retrieved from http://www.na-businesspress.com/ajmopen.html

Hans, A., & Gahlot, S. (2016). A review: Software metrics and effort estimation in aspect software development program. *International Journal of Advanced Research in Computer Science*, *7*(1). Retrieved from www.ijarcs.info

Harzl, A. (2017). Can FOSS projects benefit from integrating Kanban: a case study. *Journal of Internet Services and Applications*, *8*(1). https://doi.org/10.1186/s13174-017-0058-z

Hirschman, A. (1967). The principle of the hiding hand. *The Public Interest*, *6*. Retrieved from http://www.thepublicinterest.com

Hohl, P., Klünder, J., Bennekum, A. Van, Lockard, R., Gifford, J., Münch, J., … Schneider, K. (2018). Back to the future: origins and directions of the “Agile Manifesto” – views of the originators. *Journal of Software Engineering Research and Development*. https://doi.org/10.1186/s40411-018-0059-z

Idri, A., Amazal, F. A., & Abran, A. (2015). Analogy-based software development effort estimation: A systematic mapping and review. *Information and Software*

Technology, *58*, 206–230. https://doi.org/10.1016/j.infsof.2014.07.013

Ika, L. A. (2018). Beneficial or detrimental ignorance: The straw man fallacy of Flyvbjerg's test of Hirschman's hiding hand. *World Development*, *103*, 369–382. https://doi.org/10.1016/j.worlddev.2017.10.016

Ika, L. A., & Söderlund, J. (2016). Rethinking revisited: insights from an early rethinker. *International Journal of Managing Projects in Business*, *9*(4), 931–954. https://doi.org/10.1108/IJMPB-05-2016-0041

Ismail, M., Ismail, R., & Hamzah, N. I. (2018). Interview protocol refinement: Fine-tuning qualitative research interview questions for multi-racial populations in Malaysia. *The Qualitative Report*, *23*(11), 2700–2713. Retrieved from http://nsuworks.nova.edu/tqr/

Ivan, I., & Despa, M. L. (2016). Estimating maintenance cost for web applications. *Informatica Economica*, *20*(4/2016), 62–75. https://doi.org/10.12948/issn14531305/20.4.2016.06

Izmailov, A., Korneva, D., & Kozhemiakin, A. (2016). Effective project management with theory of constraints. *Procedia - Social and Behavioral Sciences*, *229*, 96–103. https://doi.org/10.1016/j.sbspro.2016.07.118

Jadhav, S. K., Shaga, V., & Thorat, S. D. (2017). User stories for proposed web application using agile. *International Journal of Advanced Research in Computer Science*, *8*(9), 570–574. https://doi.org//10.26483/ijarcs.v8i9.5148

Jonsen, K., Fendt, J., & Point, S. (2018). Convincing qualitative research: What constitutes persuasive writing? *Organizational Research Methods*, *21*(1), 30–67.

https://doi.org/10.1177/1094428117706533

Jørgensen, M. (2004). A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, *70*(1–2), 37–60. https://doi.org/10.1016/S0164-1212(02)00156-5

Jørgensen, M. (2014). What we do and don’t know about software development effort estimation. *Software, IEEE*, *31*(2), 37–40. https://doi.org/10.1109/MS.2014.49

Jørgensen, M. (2016). Unit effects in software project effort estimation: Work-hours gives lower effort estimates than workdays. *Journal of Systems and Software*, *117*, 274–281. https://doi.org/10.1016/j.jss.2016.03.048

Jørgensen, M., & Boehm, B. (2009). Software development effort estimation: Formal models or expert judgement? *IEEE Software*. https://10.1109/MS.2009.47

Joslin, R., & Müller, R. (2016). Identifying interesting project phenomena using philosophical and methodological triangulation. *International Journal of Project Management*, *34*(6), 1043–1056. https://doi.org/10.1016/j.ijproman.2016.05.005

Kahneman, D., & Tversky, A. (1973). On the psychology of prediction. *Psychological Review*, *80*(4), 237–251. https://doi.org/10.1037/h0034747

Kahneman, D., & Tversky, A. (1977). Intuitive prediction: Biases and corrective procedures. *Science*, 414–421. https://doi.org/10.1017/CBO9780511809477.031

Kakar, A. K. (2018). How do team conflicts impact knowledge sharing? *Knowledge Management Research and Practice*, *16*(1), 21–31. https://doi.org/10.1080/14778238.2017.1401194

Karthiekheyan, K., Ahmed, I., & Jayalakshmi, J. (2018). Pair programming for software engineering education: An empirical study. *International Arab Journal of Information Technology*, *15*(2), 246–255. Retrieved from http://www.ijarcs.info/index.php/Ijarcs

Kaur, A. (2017). Comparison of maintenance activity for effort estimation in open source software projects. *International Journal of Advanced Research in Computer Science*, *8*(5), 1535–1539. Retrieved from http://www.ijarcs.info/%0A

Kaushik, A., Tayal, D., Yadav, K., & Kaur, A. (2016). Integrating firefly algorithm in artificial neural network models for accurate software cost predictions. *Journal of Software: Evolution and Process*, *26*(12), 1172–1192. https://doi.org/10.1002/smr.1792

Kelly, K. (2017). A different type of lighting research – A qualitative methodology. *Lighting Research and Technology*, *49*(8), 933–942. https://doi.org/10.1177/1477153516659901

Khuat, T. T., & Le, H. (2016). Optimizing parameters of software effort estimation models using directed artificial bee colony algorithm. *Informatica*, *40*, 427–436. Retrieved from http://www.informatica.si/index.php/informatica

Kim, J. E., & Nembhard, D. A. (2018). Parametric empirical Bayes estimation of individual time-pressure reactivity. *International Journal of Production Research*, *56*(7), 2452–2463. https://doi.org/10.1080/00207543.2017.1380321

Kim, J. E., Nembhard, D. A., & Kim, J. H. (2016). The effects of group size and task complexity on deadline reactivity. *International Journal of Industrial Ergonomics*, *56*, 106–114. https://doi.org/10.1016/j.ergon.2016.09.011

Kirmani, M. (2017a). Agile methods for mobile application development: A comparative analysis. *International Journal of Advanced Research in Computer Science*, *8*(5). Retrieved from http://www.ijarcs.info/index.php/Ijarcs

Kirmani, M. (2017b). Software effort estimation in early stages of software development: A review. *International Journal of Advanced Research in Computer Science*, *8*(5), 1155–1159. Retrieved from www.ijarcs.info

Köhler, T., Smith, A., & Bhakoo, V. (2018). Feature topic for ORM : "Templates in qualitative research methods." *Organizational Research Methods*, *22*(1). https://doi.org/10.1177/1094428118805165

Korstjens, I., & Moser, A. (2018). Series: Practical guidance to qualitative research. Part 4: Trustworthiness and publishing. *European Journal of General Practice*, *24*(1), 120–124. https://doi.org/10.1080/13814788.2017.1375092

Kotaiah, B., & Khalil, M. A. (2017). Approaches for development of Software Projects: Agile methodology. *International Journal of Advanced Research in Computer Science*, *8*(1), 6. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84874327273&partnerID=tZOtx3y1

Krstić, M., Skorup, A., & Lapčević, G. (2018). Trends in agile innovation management. *International Review*, (3–4), 58–70. https://doi.org/10.5937/intrev1804058k

Kukreja, N., & Garg, U. (2017). Effort estimation of object oriented system using stochastic tree boosting technique. *International Journal of Advanced Research in Computer Science*, *8*(7), 91–96. https://doi.org/10.26483/ijarcs.v8i7.4246

Kulkarni, R. H., Padmanabham, P., Harshe, M., Baseer, K. K., & Patil, P. (2017).

Investigating agile adaptation for project development. *International Journal of Electrical and Computer Engineering*, *7*(3), 1278–1285. https://doi.org/10.11591/ijece.v7i3.pp1278-1285

Lee, M. J., & Rothenberger, M. A. (2015). Effort estimation factors for corrective software maintenance projects: A qualitative analysis of estimation criteria. *Journal of Information Technology Theory and Application*, *16*(2), 39–56. Retrieved from http://aisel.aisnet.org/jitta/

Lei, H., Ganjeizadeh, F., Jayachandran, P. K., & Ozcan, P. (2017). A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*, *43*, 59–67. https://doi.org/10.1016/j.rcim.2015.12.001

Lepenies, P. H. (2018). Statistical tests as a hindrance to understanding: What the controversy around the “Hiding Hand” reveals about research in the social sciences and conceals about project management. *World Development*, *103*, 360–365. https://doi.org/10.1016/j.worlddev.2017.10.017

Levitt, H. M., Motulsky, S. L., Wertz, F. J., Morrow, S. L., & Ponterotto, J. G. (2016). Recommendations for designing and reviewing qualitative research in psychology. *Qualitative Psychology*, *4*(1), 2–22. https://doi.org/http://dx.doi.org/10.1037/qup0000082

Lévy-Garboua, L., Askari, M., & Gazel, M. (2018). *Confidence biases and learning among intuitive Bayesians*. *Theory and Decision* (Vol. 84). Springer US. https://doi.org/10.1007/s11238-017-9612-1

Liao, H., & Hitchcock, J. (2018). Reported credibility techniques in higher education evaluation studies that use qualitative methods: A research synthesis. *Evaluation and Program Planning*, *68*(March), 157–165. https://doi.org/10.1016/j.evalprogplan.2018.03.005

Lima, J. E., West, G. B., Winston, B. E., & Wood, J. A. (2016). Measuring organizational cultural intelligence. *International Journal of Cross Cultural Management*, *16*(1), 9–31. https://doi.org/10.1177/1470595815615625

Lishner, D. A. (2015). A concise set of core recommendations to improve the dependability of psychological research. *Review of General Psychology*, *19*(1), 52–68. https://doi.org/10.1037/gpr0000028

Llerena, L., Rodriguez, N., Castro, J. W., & Acuña, S. T. (2019). Adapting usability techniques for application in open source software: A multiple case study. *Information and Software Technology*, *107*(October 2018), 48–64. https://doi.org/10.1016/j.infsof.2018.10.011

Løhre, E., & Jørgensen, M. (2016). Numerical anchors and their strong effects on software development effort estimates. *Journal of Systems and Software*, *116*, 49–56. https://doi.org/10.1016/j.jss.2015.03.015

Lopez-Martinez, J., Ramirez-Noriega, A., Juarez-Ramirez, R., Licea, G., & Martinez-Ramirez, Y. (2017). Analysis of Planning Poker Factors between University and Enterprise. *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 54–60. https://doi.org/10.1109/CONISOFT.2017.00014

Lorko, M., Servátka, M., & Zhang, L. (2019). Anchoring in project duration estimation. *Journal of Economic Behavior and Organization*, *162*, 49–65. https://doi.org/10.1016/j.jebo.2019.04.014

Lovallo, D., & Kahneman, D. (2003). Delusions of Success: How Optimism Undermines Executives' Decisions. *Harvard Business Review*, *81*(7), 1–15. Retrieved from https://hbr.org/

Majid, M. A. A., Othman, M., Mohamad, S. F., Lim, S. A. H., & Yusof, A. (2018). Piloting for interviews in qualitative research: Operationalization and lessons learnt. *International Journal of Academic Research in Business and Social Sciences*, *7*(4), 1073–1080. https://doi.org/10.6007/ijarbss/v7-i4/2916

Malterud, K., Siersma, V. D., & Guassora, A. D. (2016). Sample Size in Qualitative Interview Studies: Guided by Information Power. *Qualitative Health Research*, *26*(13), 1753–1760. https://doi.org/10.1177/1049732315617444

Matharu, G. S., Mishra, A., Singh, H., & Upadhyay, P. (2015). Empirical study of agile software development methodologies. *ACM SIGSOFT Software Engineering Notes*, *40*(1), 1–6. https://doi.org/10.1145/2693208.2693233

McAlpine, L. (2016). Why might you use narrative methodology? A story about narrative. *Eesti Haridusteaduste Ajakiri (Estonian Journal of Education)*, *4*(1), 32–57. https://doi.org/10.12697/eha.2016.4.1.02b

McConnell, S. (2006). *Software Estimation: Demystifying the Black Art*. Redmond WA: Microsoft Press.

McCusker, K., & Gunaydin, S. (2015). Research using qualitative, quantitative or mixed methods and choice based on the research. *Perfusion*, *30*(7), 537–542. https://doi.org/10.1177/0267659114559116

McGrath, C., Palmgren, P. J., & Liljedahl, M. (2018). Twelve tips for conducting qualitative research interviews. *Medical Teacher*, 1–5. https://doi.org/10.1080/0142159X.2018.1497149

McIntosh, M. J., & Morse, J. M. (2015). Situating and constructing diversity in semi-structured interviews. *Global Qualitative Nursing Research*, *2*, 233339361559767. https://doi.org/10.1177/2333393615597674

Mehta, S., & Kumari, S. (2016). A tool to evaluate the performance of UCP. *International Journal of Engineering Science*, *6*(7), 8193–8198. https://doi.org/10.4010/2016.1913

Meyer, B. (2018). Making Sense of Agile Methods. *IEEE Software*, *35*(2), 91–94. https://doi.org/10.1109/MS.2018.1661325

Miracle, V. A. (2016). The belmont report: The triple crown of research ethics. *Dimensions of Critical Care Nursing*, *35*(4), 223–228. https://doi.org/10.1097/DCC.0000000000000186

Miranda, E. (2017). Documentless assessments using nominal group interviews. *Software Quality Professional*, *19*(2). Retrieved from http://www.asq.org/pub/sqp/

Mirzaei, M., & Mabin, V. (2017). Agile project management and public policy development projects: A case study from New Zealand. *New Zealand Journal of Applied Business Research*, *15*(1), 59–76. Retrieved from https://www.manukau.ac.nz/

Misra, S. (2012). Agile software development practices: evolution, principles, and criticisms. *International Journal of Quality & Reliability Management*, *29*(9), 972–980. https://doi.org/10.1108/02656711211272863

Mittas, N., Papatheocharous, E., Angelis, L., & Andreou, A. S. (2015). Integrating non-parametric models with linear components for producing software cost estimations. *Journal of Systems and Software*, *99*, 120–134. https://doi.org/10.1016/j.jss.2014.09.025

Moharreri, K., Sapre, A. V., Ramanathan, J., & Ramnath, R. (2016). Cost-effective supervised learning models for software effort estimation in agile environments. *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, 135–140. https://doi.org/10.1109/COMPSAC.2016.85

Moser, A., & Korstjens, I. (2018). Series: Practical guidance to qualitative research. Part 3: Sampling, data collection and analysis. *European Journal of General Practice*, *24*(1), 9–18. https://doi.org/10.1080/13814788.2017.1375091

Müller, R., & Klein, G. (2019). Qualitative research submissions to Project Management Journal. *Project Management Journal*, *50*(1), 3–5. https://doi.org/10.1177/8756972818817625

Munawar, H., & Qureshi, M. R. J. (2015). Measuring the effect of CMMI quality standard on agile scrum model. *International Journal of Information Engineering and Electronic Business*, *7*(6), 46–52. https://doi.org/10.5815/ijieeb.2015.06.07

Naess, P., Andersen, J., Nicolaisen, M. S., & Strand, A. (2015). Forecasting inaccuracies: A result of unexpected events, optimism bias, technical problems, or strategic misrepresentation? *Journal of Transport and Land Use*, *8*(3), 39–55. https://doi.org/10.5198/jtlu.2015.719

Naik, P., & Nayak, S. (2017). Insights on research techniques towards cost estimation in software design. *International Journal of Electrical and Computer Engineering (IJECE)*, *7*(5), 2883. https://doi.org/10.11591/ijece.v7i5.pp2883-2894

National Institutes of Health. (n.d.). Guiding Principles for Ethical Research. Retrieved January 19, 2019, from https://www.nih.gov/health-information/nih-clinical-research-trials-you/guiding-principles-ethical-research

Nawaz, Z., Aftab, S., & Anwer, F. (2017). Simplified FDD process model. *International Journal of Modern Education and Computer Science*, *9*(9), 53–59. https://doi.org/10.5815/ijmecs.2017.09.06

Newby-Clark, I. R., Ross, M., Koehler, D. J., Buehler, R., & Griffin, D. (2000). People focus on optimistic scenarios and disregard pessimistic scenarios while predicting task completion times. *Journal of Experimental Psychology: Applied*, *6*(3), 171–182. https://doi.org/10.1037/1076-898X.6.3.171

Nguyen, V. D., & Nguyen, N. T. (2018). Intelligent collectives: Theory, applications, and research challenges. *Cybernetics and Systems*, *0*(0), 1–19. https://doi.org/10.1080/01969722.2017.1418254

Orange, A. (2016). Encouraging reflexive practices in doctoral students through research journals. *The Qualitative Report*, *21*(12), 2176. Retrieved from http://www.nova.edu/ssss/QR/index.html

Osman, H., & Musa, M. (2016). A survey of agile software estimation methods.

International Journal of Computer Science and Telecommunications, *7*(3), 38–42. Retrieved from http://www.ijcst.org/Volume7/Issue3/p6_7_3.pdf

Osmanbegović, E., Suljić, M., & Agić, H. (2017). A review of estimation of software products development cost. *Ekonomski Vjesnik/Econviews-Review of Contemporary Business, Entrepreneurship and Economic Issues*, *3*, 195–208. Retrieved from https://hrcak.srce.hr/ojs/index.php/ekonomski-vjesnik/article/view/4108

Ozierańska, A., Skomra, A., Kuchta, D., & Rola, P. (2016). The critical factors of scrum implementation in IT project – the case study. *Journal of Economics and Management*, *25*(3), 79–96. https://doi.org/10.22367/jem.2016.25.06

Parent, M. (2019). Unbiasing information technology decisions. *Organizational Dynamics*. https://doi.org/10.1016/j.orgdyn.2019.02.001

Perkusich, M., Gorgônio, K. C., Almeida, H., & Perkusich, A. (2017). Assisting the continuous improvement of scrum projects using metrics and bayesian networks. *Journal of Software: Evolution and Process*, *29*(6), 1–18. https://doi.org/10.1002/smr.1835

Peticca-Harris, A., DeGama, N., & Elias, S. R. S. T. A. (2016). A dynamic process model for finding informants and gaining access in qualitative research. *Organizational Research Methods*, *19*(3), 376–401. https://doi.org/10.1177/1094428116629218

Phillippi, J., & Lauderdale, J. (2018). A guide to field notes for qualitative research: Context and conversation. *Qualitative Health Research*, *28*(3), 381–388. https://doi.org/10.1177/1049732317697102

Phillips, M., & Lu, J. (2018). A quick look at NVivo. *Journal of Electronic Resources*

Librarianship, *30*(2), 104–106. https://doi.org/10.1080/1941126x.2018.1465535

Pinto, J. K. (2013). Lies, damned lies, and project plans: Recurring human errors that can ruin the project planning process. *Business Horizons*, *56*(5), 643–653. https://doi.org/10.1016/j.bushor.2013.05.006

Polonioli, A. (2016). Adaptive rationality, biases, and the heterogeneity hypothesis. *Review of Philosophy and Psychology*, *7*(4), 787–803. https://doi.org/10.1007/s13164-015-0281-0

Ponelis, S. R. (2015). Using interpretive qualitative case studies for exploratory research in doctoral studies: A case of information systems research in small and medium enterprises. *International Journal of Doctoral Studies*, *10*, 535–550. https://doi.org/10.28945/2339

Pospieszny, P., Czarnacka-Chrobot, B., & Kobylinski, A. (2018). An effective approach for software project effort and duration estimation with machine learning algorithms. *Journal of Systems and Software*, *137*, 184–196. https://doi.org/10.1016/j.jss.2017.11.066

Prakash, B., & Viswanathan, V. (2017). A survey on software estimation techniques in traditional and agile development models. *Indonesian Journal of Electrical Engineering and Computer Science*, *7*(3), 867–876. https://doi.org/10.11591/ijeecs.v7.i3.pp867-876

Prater, J., Kirytopoulos, K., & Ma, T. (2017). Optimism bias within the project management context: A systematic quantitative literature review. *International Journal of Managing Projects in Business*, *10*(2), 370–385.

https://doi.org/10.1108/IJMPB-07-2016-0063

Qi, F., Jing, X. Y., Zhu, X., Xie, X., Xu, B., & Ying, S. (2017). Software effort estimation based on open source projects: Case study of Github. *Information and Software Technology*, *92*, 145–157. https://doi.org/10.1016/j.infsof.2017.07.015

Rahi, S. (2017). Research design and methods: A systematic review of research paradigms, sampling issues and instruments development. *International Journal of Economics & Management Sciences*, *06*(02). https://doi.org/10.4172/2162-6359.1000403

Rahikkala, J., Leppänen, V., Ruohonen, J., & Holvitie, J. (2015). Top management support in software cost estimation: A study of attitudes and practice in Finland. *International Journal of Managing Projects in Business*, *8*(3), 513–532. https://doi.org/10.1108/IJMPB-11-2014-0076

Rai, A., Gupta, G. P., & Kumar, P. (2017). Estimation of software development efforts using improved delphi technique: A novel approach. *International Journal of Applied Engineering Research*, *12*(12), 3228–3236. Retrieved from http://www.ripublication.com/ijaer.htm

Ramirez-Noriega, A., Juarez-Ramirez, R., Navarro, R., & Lopez-Martinez, J. (2016). Using Bayesian networks to obtain the task's parameters for schedule planning in scrum. *Proceedings - 2016 4th International Conference in Software Engineering Research and Innovation, CONISOFT 2016*, (1), 167–174. https://doi.org/10.1109/CONISOFT.2016.33

Raslan, A. T., Darwish, N. R., & Hefny, H. A. (2015). Towards a fuzzy based framework for effort estimation in agile software development. *International Journal of Computer Science and Information Security*, *13*(1), 37. Retrieved from http://search.proquest.com/openview/0ffd04c753ca702dbd632daf9f7ffb81/1?pq-origsite=gscholar&cbl=616671

Rath, S. K., Acharya, B. P., & Satapathy, S. M. (2016). Early stage software effort estimation using random forest technique based on use case points. *IET Software*, *10*(1), 10–17. https://doi.org/10.1049/iet-sen.2014.0122

Riccobono, F., Bruccoleri, M., & Größler, A. (2016). Groupthink and project performance: The influence of personal traits and interpersonal yies. *Production and Operations Management*, *25*(4), 609–629. https://doi.org/10.1111/poms.12431

Rijwani, P., & Jain, S. (2016). Enhanced Software Effort Estimation Using Multi Layered Feed Forward Artificial Neural Network Technique. *Procedia Computer Science*, *89*, 307–312. https://doi.org/10.1016/j.procs.2016.06.073

Rimando, M., Brace, A. M., Namageyo-Funa, A., Parr, T. L., & Sealy, D. A. (2015). Data collection challenges and recommendations for early career researchers. *The Qualitative Report*, *20*(12), 2025–2036. Retrieved from http://nsuworks.nova.edu/tqr/vol20/iss12/8

Røddesnes, S., Faber, H. C., & Jensen, M. R. (2019). NVivo courses in the library: Working to create the library services of tomorrow, *11*(1), 27–38. https://doi.org/ 10.15845/noril.v11i1.2762

Room, G. (2018). The hiding hand: A rejoinder to Flyvbjerg on Hirschman. *World Development*, *103*, 366–368. https://doi.org/10.1016/j.worlddev.2017.10.015

Roulston, K. (2018). Qualitative interviewing and epistemics. *Qualitative Research*, *18*(3), 322–341. https://doi.org/10.1177/1468794117721738

Roy, M. M., Burns, T., & Radzevick, J. R. (2019). Unpacking, summing and anchoring in retrospective time estimation. *Acta Psychologica*, *192*(November 2018), 153–162. https://doi.org/10.1016/j.actpsy.2018.11.012

Sadaf, S., Iqbal, S., Saba, A., & Mohsin, M. M. (2017). An extended adaptive process model for agile software development methodology. *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, 1373–1378. https://doi.org/10.1109/ICICICT1.2017.8342770

Sambare, T. (2017). Agility: The need of an hour for software industry. *International Journal of Advanced Research in Computer Science*, *8*(0976). https://doi.org/http://dx.doi.org/10.26483/ijarcs.v8i9.4886

Saravanan, K., Floyd, R. W., Mcllroy, D., Morris, C., Boehm, B., Methodo, C., … North, D. (2017). Systems development methodologies: Conceptual study. *Indian Journal of Scientific Research*, *14*(1), 27–37. Retrieved from http://www.ijsr.in

Sarstedt, M., Bengart, P., Shaltoni, A. M., & Lehmann, S. (2018). The use of sampling methods in advertising research: a gap between theory and practice. *International Journal of Advertising*, *37*(4), 650–663. https://doi.org/10.1080/02650487.2017.1348329

Saunders, M. N. K., & Townsend, K. (2016). Reporting and justifying the number of interview participants in organization and workplace research. *British Journal of Management*, *27*(4), 836–852. https://doi.org/10.1111/1467-8551.12182

Sehra, S. K., Brar, Y. S., Kaur, N., & Sehra, S. S. (2017). Research patterns and trends in software effort estimation. *Information and Software Technology*, *0*, 1–21. https://doi.org/10.1016/j.infsof.2017.06.002

Sehra, S. K., Brar, B. Y., & Kaur, N. (2016). Predominant factors influencing software effort estimation. *International Journal of Computer Science and Information Security*, *14*(7), 107–110. Retrieved from https://sites.google.com/site/ijcsis/

Seixas, B. V., Smith, N., & Mitton, C. (2018). The Qualitative Descriptive Approach in International Comparative Studies: Using Online Qualitative Surveys. *International Journal of Health Policy and Management*, *7*(9), 778–781. https://doi.org/10.15171/ijhpm.2017.142

Shannon-Baker, P. (2016). Making paradigms meaningful in mixed methods research. *Journal of Mixed Methods Research*, *10*(4), 319–334. https://doi.org/10.1177/1558689815575861

Shawky, D. M., Salwa, K., & El-Hafiz, A. (2016). Characterizing software development method using metrics. *Journal of Software: Evolution and Process*, *28*, 82–96. https://doi.org/10.1002/smr.176

Shekhar, S., & Kumar, U. (2016). Review of various software cost estimation techniques. *International Journal of Computer Applications*, *141*(11), 31–34. Retrieved from http://www.ijcaonline.org/

Shepperd, J. A., Waters, E. A., Weinstein, N. D., & Klein, W. M. P. (2015). A primer on unrealistic optimism. *Current Directions in Psychological Science*, *24*(3), 232–237. https://doi.org/10.1177/0963721414568341

Shepperd, M., Mair, C., & Jørgensen, M. (2018). An experimental evaluation of a de-biasing intervention for professional software developers. In *Proceedings of the ACM Symposium on Applied Computing* (pp. 1510–1517). https://doi.org/10.1145/3167132.3167293

Shida, T., & Tsuda, K. (2017). A study of software estimation factors extracted using covariance structure analysis. *Procedia Computer Science*, *112*, 1378–1387. https://doi.org/10.1016/j.procs.2017.08.053

Shmueli, O., Pliskin, N., & Fink, L. (2016). Can the outside-view approach improve planning decisions in software development projects? *Information Systems Journal*, *26*(4), 395–418. https://doi.org/10.1111/isj.12091

Shmueli, O., & Ronen, B. (2017). Excessive software development: Practices and penalties. *International Journal of Project Management*, *35*(1), 13–27. https://doi.org/10.1016/j.ijproman.2016.10.002

Shollig, Widodo, A. P., Sutanto, T., & Subriadi, A. P. (2016). A model to determine cost estimation for software development projects of small and medium scales using use case points. *Journal of Theoretical and Applied Information Technology*, *85*(1), 1–8. Retrieved from http://www.jatit.org

Singh, A., & Pandey, D. (2017). Implementation of requirement engineering in extreme programming and scrum. *International Journal of Advanced Research in Computer Science*, *8*(5), 621–625. Retrieved from http://www.ijarcs.info/index.php/Ijarcs

Solinski, A., & Petersen, K. (2016). Prioritizing agile benefits and limitations in relation to practice usage. *Software Quality Journal*, *24*(2), 447–482.

https://doi.org/10.1007/s11219-014-9253-3

Soni, D., & Kohli, P. J. (2017). Cost estimation model for web applications using agile software development methodology. *Pertanika Journal of Science and Technology*, *25*(3), 931–938. Retrieved from http://www.pertanika.upm.edu.my/index%20-%20JST.htm

Srivastava, P., & Jain, S. (2017). A leadership framework for distributed self-organized scrum teams. *Team Performance Management*, *23*(5–6), 293–314. https://doi.org/10.1108/TPM-06-2016-0033

Staats, B. R., Milkman, K. L., & Fox, C. R. (2012). The team scaling fallacy: Underestimating the declining efficiency of larger teams. *Organizational Behavior and Human Decision Processes*, *118*(2), 132–142. https://doi.org/10.1016/j.obhdp.2012.03.002

Stake, R. (2006). *Multiple Case Study Analysis.* New York, NY: Guilford Press. Retrieved from https://books.google.com/books?hl=en&lr=&id=rQWT5aDHiZYC&oi=fnd&pg=PT21&dq=Stake,+R.+E&ots=IGdSIyExzo&sig=q9JLy-EhctE0cWwmz_rp8HOqUC4#v=onepage&q=Stake%2C R. E&f=false

Starcher, R. L., Dzubinski, L. M., & Sanchez, J. N. (2018). Rigorous missiological research using qualitative inquiry. *Missiology: An International Review*, *46*(1), 50–66. https://doi.org/10.1177/0091829617741911

Sting, F. J., Loch, C. H., & Stempfhuber, D. (2015). Accelerating projects by encouraging help. *MIT Sloan Management Review*, *56*(3), 1–9. Retrieved from http://sloanreview.mit.edu/

Stoica, M., Ghilic-Micu, B., Mircea, M., & Uscatu, C. (2016). Analyzing agile development - from waterfall style to scrumban. *Informatica Economica*, *20*(4/2016), 15–25. https://doi.org/10.12948/issn14531305/20.4.2016.02

Stokes, Y., Vandyk, A., Squires, J., Jacob, J. D., & Gifford, W. (2019). Using Facebook and LinkedIn to recruit nurses for an online survey. *Western Journal of Nursing Research*, *41*(1), 96–110. https://doi.org/10.1177/0193945917740706

Strasser, A. (2017). Delphi method variants in information systems research: Taxonomy development and application. *Electronic Journal of Business Research Methods*, *15*(2), 120–133. Retrieved from http://www.ejbrm.com/index.htm

Surmiak, A. (2018). Confidentiality in qualitative research involving vulnerable participants: Researchers' perspectives. *Forum Qualitative Sozialforschung*, *19*(3). https://doi.org/10.17169/fqs-19.3.3099

Tanner, M., & Dauane, M. (2017). The use of Kanban to alleviate collaboration and communication challenges of global software development. *Issues in Informing Science & Information Technology*, *14*, 177. Retrieved from https://www.informingscience.org/Journals/IISIT/Overview

Tanveer, B. (2017). Guidelines for utilizing change impact analysis when estimating effort in agile software development. In *ACM* (pp. 252–257). https://doi.org/10.1145/3084226.3084284

Tarwani, S., & Chug, A. (2016). Agile Methodologies in Software Maintenance: A Systematic Review. *Informatica 40*, *40*, 421. Retrieved from http://www.informatica.si/index.php/informatica/article/view/1182/927

Taylor, K. J. (2016). Adopting agile software development: the project manager experience. *Information Technology and People*, *29*(4), 670–687. https://doi.org/10.1108/ITP-02-2014-0031

Tenenboim, E., & Shiftan, Y. (2018). Accuracy and bias of subjective travel time estimates. *Transportation*, *45*(3), 945–969. https://doi.org/10.1007/s11116-016-9757-8

The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. (1979). The Belmont Report | HHS.gov. Retrieved February 20, 2019, from https://www.hhs.gov/ohrp/regulations-and-policy/belmont-report/read-the-belmont-report/index.html

Thomas, K. E., & König, C. J. (2018). Knowledge of previous tasks: Task similarity influences bias in task duration predictions. *Frontiers in Psychology*, *9*(MAY), 1–14. https://doi.org/10.3389/fpsyg.2018.00760

Tomczak, P., & Traczyk, J. (2017). The mechanism of non-numerical anchoring heuristic based on magnitude priming: Is it just the basic anchoring effect in disguise? *Polish Psychological Bulletin*, *48*(3), 401–410. https://doi.org/10.1515/ppb-2017-0046

Tong, A., & Dew, M. A. (2016). Qualitative research in transplantation. *Transplantation*, *100*(4), 710–712. https://doi.org/10.1097/tp.0000000000001117

Torrecilla-Salinas, C. J., Sedeño, J., Escalona, M. J., & Mejías, M. (2015). Estimating, planning and managing Agile Web development projects under a value-based perspective. *Information and Software Technology*, *61*, 124–144.

https://doi.org/10.1016/j.infsof.2015.01.006

Tversky, A., & Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science*, *185*(November), 0–2. https://doi.org/10.1080/10548400802156836

Twining, P., Heller, R. S., Nussbaum, M., & Tsai, C. C. (2017). Some guidance on conducting and reporting qualitative studies. *Computers and Education*, *106*, A1–A9. https://doi.org/10.1016/j.compedu.2016.12.002

Urbanek, T., Kolcavova, A., & Kuncar, A. (2017). Inferring productivity factor for use case point method. *Annals of DAAAM and Proceedings of the International DAAAM Symposium*, 597–602. https://doi.org/10.2507/28th.daaam.proceedings.084

Usländer, T. (2016). Agile Service-oriented Analysis and Design of Industrial Internet Applications. *Procedia CIRP*, *57*, 219–223. https://doi.org/10.1016/j.procir.2016.11.038

Usman, M., Börstler, J., & Petersen, K. (2017). An effort estimation taxonomy for agile software development. *International Journal of Software Engineering and Knowledge Engineering*, *27*(04), 641–674. https://doi.org/10.1142/S0218194017500243

Usman, M., Britto, R., Damm, L. O., & Börstler, J. (2018). Effort estimation in large-scale software development: An industrial case study. *Information and Software Technology*, *99*(April 2017), 21–40. https://doi.org/10.1016/j.infsof.2018.02.009

Vallon, R., José, B., Prikladnicki, R., & Grechenig, T. (2018). Systematic literature review on agile practices in global software development. *Information and Software Technology*, *96*(April 2017), 161–180. https://doi.org/10.1016/j.infsof.2017.12.004

van Baalen, S. (2018). ‘Google wants to know your location’: The ethical challenges of fieldwork in the digital age. *Research Ethics*, *14*(4), 1–17. https://doi.org/10.1177/1747016117750312

van Vliet, H., & Tang, A. (2016). Decision making in software architecture. *Journal of Systems and Software*, *117*, 638–644. https://doi.org/10.1016/j.jss.2016.01.017

Venkatesh, V., Brown, S. A., & Sullivan, Y. W. (2016). Guidelines for conducting mixed-methods research: An extension and illustration. *Journal of the Association for Information Systems*, *17*(7), 435–494. Retrieved from https://aisel.aisnet.org/jais/

Vicary, S., Young, A., & Hicks, S. (2017). A reflective journal as learning process and contribution to quality and validity in interpretative phenomenological analysis. *Qualitative Social Work*, *16*(4), 550–565. https://doi.org/10.1177/1473325016635244

Vyas, M., Bohra, A., Lamba, C. S., & Vyas, A. (2018). A Review on Software Cost and Effort Estimation Techniques for Agile Development Process. *International Journal of Recent Research Aspects*, *5*(1), 1–5. Retrieved from https://www.ijrra.net/

Wiese, J., Buehler, R., & Griffin, D. (2016). Backward planning: Effects of planning direction on predictions of task completion time. *Judgment and Decision Making*, *11*(2), 147–167. Retrieved from http://journal.sjdm.org

Wilson, V. (2016). Research methods: sampling. *Evidence Based Library and Information Practice*, *11*(1). https://doi.org/10.18438/b8wc8n

Windsong, E. A. (2018). Incorporating intersectionality into research design: An example using qualitative interviews. *International Journal of Social Research Methodology*, *21*(2), 135–147. https://doi.org/10.1080/13645579.2016.1268361

Xerri, D. (2017). Two methodological challenges for teacher-researchers: Reflexivity and trustworthiness. *The Clearing House: A Journal of Educational Strategies, Issues and Ideas*, *91*(1), 37–41. https://doi.org/10.1080/00098655.2017.1371549

Yamini, S., & Marathe, R. R. (2018). Mathematical model to mitigate planning fallacy and to determine realistic delivery time. *IIMB Management Review*, 242–257. https://doi.org/10.1016/j.iimb.2018.05.003

Yang, C., Liang, P., & Avgeriou, P. (2018). Assumptions and their management in software development: A systematic mapping study. *Information and Software Technology*, *94*(December 2016), 82–110. https://doi.org/10.1016/j.infsof.2017.10.003

Yazan, B. (2015). Three approaches to case study methods in education: Yin, Merriam, and Stake. *The Qualitative Report*, *20*(2), 134–152. Retrieved from http://www.nova.edu/ssss/QR/QR20/2/yazan1.pdf

Yilmaz, K. (2013). Comparison of quantitative and qualitative research traditions: epistemological, theoretical and methodological differences. *European Journal of Education*, *48*(2), 311–325. https://doi.org/doi:10.1111/ejed.12014

Yin, R. K. (2014). *Case Study Research: Design & Methods* (5th ed.). Thousand Oaks, CA: Sage Publications. https://doi.org/10.1007/BF01103312

Yoshigami, K., Tsunoda, M., Yamada, Y., & Kusumoto, S. (2017). Should function point elements be used to build prediction models? *Proceedings - 8th IEEE International Workshop on Empirical Software Engineering in Practice, IWESEP 2017*, 41–46.

https://doi.org/10.1109/IWESEP.2017.18

Younas, M., Ghani, I., Jawawi, D. N. A., & Khan, M. M. (2016). A Framework for Agile Development in Cloud Computing Environment ☆. *Journal of Internet Computing and Services (JICS)*, *17*(5), 67–74. https://doi.org/10.7472/jksii.2016.17.5.67

Zadeh, L. A. (2015). Fuzzy logic - A personal perspective. *Fuzzy Sets and Systems*, *281*, 4–20. https://doi.org/10.1016/j.fss.2015.05.009

Zahraoui, H., & Idrissi, M. A. J. (2015). Adjusting story points calculation in scrum effort & time estimation. *2015 10th International Conference on Intelligent Systems: Theories and Applications, SITA 2015*. https://doi.org/10.1109/SITA.2015.7358400

Zare, F., Zare, H., & Fallahnezhad, M. S. (2016). Software effort estimation based on the optimal Bayesian belief network. *Applied Soft Computing Journal*, *49*, 968–980. https://doi.org/10.1016/j.asoc.2016.08.004

Zhang, J., Jia, S., & Diaz, E. (2018). Dynamic monitoring and control of a critical chain project based on phase buffer allocation. *Journal of the Operational Research Society*, *69*(12), 1966–1977. https://doi.org/10.1080/01605682.2017.1415641

Zhu, D., Li, X., Yang, S., & Xie, X. (2019). More accurate or less accurate: How does maximization orientation affect task completion predictions? *Personality and Individual Differences*, *137*(April 2018), 173–183. https://doi.org/10.1016/j.paid.2018.08.025

Zulfikar, T., & Mujiburrahman. (2018). Understanding own teaching: becoming reflective teachers through reflective journals. *Reflective Practice*, *19*(1), 13. https://doi.org/10.1080/14623943.2017.1295933

Appendix A: Participant Prescreen Questions

1. Do you or your team provide software development effort estimates?
2. How many years experience do you have providing software development effort estimates?
3. Do you currently use an effort estimation strategy for software development changes that is considered accurate by the project managers or product managers?
4. Is your effort estimation method a formal process?

Appendix B: Participant Invitation E-mail

Kevin Roark
Doctoral Candidate at Walden University
Kevin.Roark@waldenu.edu
[telephone number redacted]

Date:

Dear Participant

I am a doctoral student at Walden University working on my doctoral project for completion of my doctoral degree. My study will be to explore the strategies used by agile software development professionals to provide software development project managers with accurate software development effort estimations.

You have been selected as a potential participant in my study based on your knowledge and use of successful strategies in providing accurate estimations of software development effort. The study will require that I meet with you to conduct an interview and review non-proprietary information about estimation strategies you use. The data that I will report on and publish will not disclose any information that would uniquely identify you or your company. Participation in this study is voluntary and will include an interview that will last about one hour.

Your consideration to participate in this study is appreciated. If you can participate, please respond to me at kevin.roark@waldenu.edu. Your participation in this study will help other software development professional and organizations to understand successful estimation strategies. The results of this study will provide effective estimation strategies in software development planning.

Thank you for your consideration. Please feel free to reach out to me by email or phone should you have any questions.

Sincerely
Kevin Roark
Walden University
Doctoral Candidate
[telephone number redacted]

Appendix C: Interview Protocol

1. Introduce myself to the participant and describe my role as a researcher
2. Briefly discuss the objectives of the study and why I selected them to interview
3. Provide the participant the Informed Consent form to sign and discuss the purpose of informed consent.
4. I will remind the participant that participation in the study is voluntary and the right to withdraw from the study by sending me an email before my member checking process.
5. I will remind the participants that the recording, personal notes, and the transcription will not include any personally identifiable information and that I will maintain their anonymity and preserve their confidentiality.
6. I will remind the participant that the interview will be recorded for transcription purposes.
7. Ask the participant if they have any questions before the interview begins.
8. Start the recording and begin asking the participant the interview questions in order.
9. Ask the participant if there are any organization process or procedure documents that describe or detail relevant information regarding this study.
10. Thank the participant for their participation in the study
11. Inform the participant that I will be contacting them as a followup on my interview to ensure I have interpreted their interview data correctly.