2017

Collaboration Strategies to Reduce Technical Debt

Jeffrey Allen Miko
Walden University

Walden University

College of Management and Technology

This is to certify that the doctoral study by

Jeffrey Allen Miko

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.

Review Committee
Dr. Jon McKeeby, Committee Chairperson, Information Technology Faculty
Dr. Steven Case, Committee Member, Information Technology Faculty
Dr. Gail Miles, University Reviewer, Information Technology Faculty

Chief Academic Officer
Eric Riedel, Ph.D.

Walden University
2017

Abstract

Collaboration Strategies to Reduce Technical Debt

by

Jeffrey Allen Miko

MS, University of Illinois-Springfield, 2013

BS, Thomas Edison State University, 2012

Doctoral Study Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Information Technology

Walden University

October 2017

Abstract

Inadequate software development collaboration processes can allow technical debt to accumulate increasing future maintenance costs and the chance of system failures. The purpose of this qualitative case study was to explore collaboration strategies software development leaders use to reduce the amount of technical debt created by software developers. The study population was software development leaders experienced with collaboration and technical debt at a large health care provider in the state of California. The data collection process included interviews with 8 software development leaders and reviewing 19 organizational documents relating to software development methods. The extended technology acceptance model was used as the conceptual framework to better understand the social and cognitive influences on the perceived usefulness of collaboration in reducing technical debt. An inductive analysis of the data was used for coding, triangulation, and identifying themes related to the use of collaboration strategies to reduce technical debt. Prominent themes included using collaboration at all stages of development, using continuous verification processes, promoting a participatory culture, and using tools to support distributed teams. The study findings showed an environment that promotes collaboration, a culture that encourages participation, and accessibility to collaborative tools that may reduce technical debt in software projects. The results of this study may contribute to positive social change by demonstrating how individuals with diverse backgrounds and different perspectives can work together to improve critical software that people depend on every day.

Collaboration Strategies to Reduce Technical Debt

by

Jeffrey Allen Miko

MS, University of Illinois-Springfield, 2013

BS, Thomas Edison State University, 2012

Doctoral Study Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Information Technology

Walden University

October 2017

Dedication

I dedicate this work to my parents, Mr. and Mrs. Joseph and Sandra Miko, who have supported me in every endeavor in my life.

Acknowledgments

I would like to thank my family for their support and encouragement in my pursuit of education starting many years ago. I would like to thank my committee chair, Dr. Jon McKeeby, who provided tremendous support, guidance, and encouragement throughout my doctoral journey. I would like to thank you for your patience and support. Thank you to my second committee member, Dr. Steven Case, for his valuable feedback, helping me better understand the research process, and sharing his wealth of knowledge. I would like to thank Dr. Gail Miles for her feedback helping me make my study more complete. I would like to thank the DIT student community for their support and encouragement.

Table of Contents

List of Tables

List of Figures

Section 1: Foundation of the Study

Background of the Problem

Collaboration is an important facet of software development considering three of the four daily activities performed by software developers involve collaborative tasks that can affect the success of software teams (Dullemond, Van Gameren, & Van Solingen, 2014). Software development often relies on different teams working in different geographic locations, so project success often depends on effective collaboration among team projects (Sundaramoorthy & Bharathi, 2016). Software development organizations that strive to improve collaboration throughout the software life cycle are more successful than those that do not improve collaboration (Lesser & Ban, 2016).

Technical debt is the increase in costs associated with maintaining or enhancing a system resulting from convenient shortcuts taken during the development of the system that did not align with industry best practices (Ampatzoglou, Ampatzoglou, Chatzigeorgiou, & Avgeriou, 2015). Poor collaboration strategies make it easier for technical debt to accumulate without detection, and this lack of awareness makes it easier for individuals to make additional decisions incurring more debt (Tom, Aurum, & Vidgen, 2013). Software development best practices requiring collaboration such as requirements gathering, design reviews, code reviews, and mentoring may increase technical debt due to ineffective collaboration activities. My study focused on the technical debt that originates from software development teams due to ineffective collaboration strategies. I explored the characteristics of collaboration and examined strategies that can be used to minimize technical debt.

Problem Statement

Technical debt refers to future maintenance costs accumulated by software development teams taking shortcuts in the development processes leading to 50% decreases in long-term customer satisfaction (Ramasubbu & Kemerer, 2014). Technical debt in enterprise software systems increases the chance of system failures by up to 62% (Ramasubbu & Kemerer, 2015) with software systems containing $3.61 of technical debt principal per line of code (Li, Liang, Avgeriou, Guelfi, & Ampatzoglou, 2014). The general IT problem was that poor software development collaboration processes create technical debt in software systems requiring additional maintenance costs to provide software updates. The specific IT problem was that some software development leaders lack collaboration strategies to reduce the amount of technical debt created by software developers.

Purpose Statement

The purpose of this qualitative case study was to explore collaboration strategies software development leaders use to reduce the amount of technical debt created by software developers. The population for this study included senior software development leaders from a large health care provider in the state of California. The population was experienced with the phenomenon of technical debt and were involved in software development collaboration. These senior software development leaders participated in semistructured interviews to identify how the collaboration of software development teams affects technical debt. The implications for positive social change include the potential to increase the reliability of communal software systems and reduce the

economic burden of software on society by improving the collaboration among software development professionals.

Nature of the Study

The qualitative methodology was the research method selected for this study. A qualitative method focuses on the participants' perspectives, meanings, and subjective views allowing the researcher to view the phenomenon from the viewpoint of the participants (Yilmaz, 2013). A qualitative method allows the researcher to be a key instrument of the data collection process, to analyze data inductively and recursively, to develop a complex picture of the issue being studied, and to reflect on his or her role in the study (Yilmaz, 2013). My research involved collecting and analyzing data from interviews and software development artifacts pertaining to standards, processes, methodologies, and collaboration to develop a comprehensive understanding of the phenomenon I studied. In a quantitative study, the researcher uses a deductive approach using a theory that relates to the topic under study, develops one or more hypotheses based on this theory, and tests the hypotheses with data using statistical procedures (Barczak, 2015). A quantitative approach focuses on proving or disproving hypotheses through investigation of relationships between independent and dependent variables (Barczak, 2015). A quantitative method was not appropriate for this study because there was no testing of a theory or hypotheses, there were no independent or dependent variables, and there was no collection of numeric data for statistical testing. Because a mixed-methods research method incorporates quantitative data collection and statistical analysis (Hayes, Bonner, & Douglas, 2013), it was not appropriate for this study.

A single exploratory case study design was the most appropriate design for this study. A case study is conducted to develop a detailed interpretation of a specific case or multiple cases by studying an event, program, or activity (Wynn & Williams, 2012). A qualitative case study design improves a researcher's understanding of a phenomenon through a comprehensive examination to investigate a multifaceted phenomenon in a real-world setting asking *how* or *what* questions (Cronin, 2014; Wynn & Williams, 2012). Ethnographic studies focus on describing cultures and social behaviors (Walker, 2012) to gain a better understanding of participants' social behaviors within their culture (Cruz & Higginbottom, 2013). My study did not focus on the cultural or social behaviors of participants. A narrative research design is used to explore the life of an individual (Walker, 2012). My study focused on collaboration between multiple individuals, and studying a single individual would not have yielded the appropriate data to answer my research question. A phenomenological research design is used to study the human experience from the perspective of the participants living through the phenomenon (Hanson, Balmer, & Giardino, 2011). My study did not focus on the lived experiences of participants, so a phenomenological research design was not appropriate. The purpose of this study was to explore collaboration strategies software development teams need to minimize technical debt.

Research Question

The overarching research question for this study was the following: What collaboration strategies do software development leaders use to minimize technical debt created by their software developers?

Demographic Questions

1. What is your current position and role?
2. How long have you been in your current position?
3. How many years of experience do you have in software development?
4. What degrees and industry certifications do you possess?

Interview Questions

1. How would you describe collaboration and its purpose in software development? What have been the benefits of collaboration to your software development team and your organization? These questions will inform me as to the job relevance and the perceived usefulness of collaboration.
2. What are the methods and tools your team uses to facilitate collaboration? How would you describe the usefulness of those methods and tools? How easy have those methods and tools been to use? These questions will inform me as to the perceived usefulness and perceived ease of use of the collaboration method and tools used by the software development team.
3. What collaboration strategies has your team used to ensure programming logic meets requirements, software designs are accurate, and programming code is free of defects? How would you describe the usefulness of those strategies to the overall success of your current projects? How easy were those strategies to implement? These questions will inform me as to the job relevance, output quality, and perceived usefulness of collaboration strategies. The questions will also inform me as to the perceived ease of use of these strategies.

4. What collaboration strategies has your team used to ensure team members follow your software development processes, policies, and best practices? How would you describe the usefulness of those strategies in preventing future bug fixing, code refactoring, and design changes? How easy were those strategies to implement? These questions will inform me of the voluntariness and compliance with social influences, which indicate the team members' intention to use collaboration.
5. How would you describe technical debt and its effects on software development projects? What have been the largest sources of technical debt in your organization? How has your team managed technical debt? These questions will inform me as to the behavioral use and perceived usefulness of collaboration strategies.
6. How would you describe the collaboration strategies your team has used to identify, prevent, and reduce technical debt? How would you describe the usefulness of those strategies in managing your technical debt? Which strategies were the most useful in minimizing technical debt? Which strategies were the easiest to implement? These questions will inform me as to the perceived usefulness, intentions to use, and usage behaviors of collaboration strategies.
7. Explain why you might have made changes to your team's collaboration strategies in the past and how these changes affected your team's ability to minimize or reduce technical debt in projects? These questions will inform me

as to the perceived usefulness, intentions to use, and usage behaviors of collaboration strategies.

Conceptual Framework

Venkatesh and Davis's (2000) extension of the technology acceptance model (TAM2) formed the basis for the conceptual framework for this study. Davis (1986) first introduced the concept of the technology acceptance model (TAM) as part of a doctoral dissertation, and refined the theory a few years later (Davis, Bagozzi, & Warshaw, 1989). The main tenet of the TAM is that a user's or group's acceptance of technology is dependent on the perceived usefulness and perceived ease of use of the technology (Conrad, 2013). The TAM2 expands on the original TAM by adding social influence processes and cognitive instrumental processes as determinants of the perceived usefulness and perceived ease of use technology (Venkatesh & Davis, 2000).

The intent of this study was to explore collaboration strategies that software development leaders employ to minimize technical debt. These strategies include the use of tools, technologies, and technical processes. The TAM2 provided a theoretical basis to study the adoption and effectiveness of various collaboration strategies at reducing technical debt through investigation of the social and cognitive processes that determine the perceived usefulness and ease of use of these collaboration strategies. The TAM2 provided a lens to examine the subjective norms, voluntariness, and image (see Venkatesh & Davis, 2000) of software development teams to improve the understanding of social influences on collaboration strategies. The TAM2 also provided a method to understand the effects of job relevance, output quality, and result demonstrability (see

Venkatesh & Davis, 2000) of software developers on the cognitive decision-making processes regarding the choice of collaboration strategies.

Definition of Terms

Agile software development: Agile software development uses incremental, iterative work cadences to assist teams in responding to unpredictability in projects where solutions evolve through collaboration between self-organizing, cross-functional teams (Cubric, 2013). Agile methodologies are an alternative to a waterfall, or traditional sequential development.

Code review: A code review is a collaboration between the authors of programming code and those reviewing the code to identify defects, improve the maintainability of the code, and share knowledge among team members (McIntosh, Kamei, Adams, & Hassan, 2015). Code reviews facilitate identification of violations of software development best practices to improve software quality (Foganholi, Garcia, Eler, Correia, & Junior, 2015).

Code smell: A code smell, or coding violation, indicates source code that does not follow the principles of object-oriented programming or design (Foganholi et al., 2015). A class having more than one purpose is an example of violating object-oriented best practices.

Defect density: Defect density is a measurement of code quality using the ratio of defects per lines of source code (di Bella et al., 2013). Defect density provides a normalized and comparable method to measure code quality among varying code sizes (di Bella et al., 2013).

Unstructured code: Unstructured code refers to large sections of code where programming logic and functionality are contained in the same section and separated or modularized, which leads to readability, redundancy, and maintenance problems (Hall, Min, Bowes, & Yi, 2014). Unstructured code is difficult to understand and may contain duplicate or redundant code (Hall et al., 2014).

Assumptions, Limitations, and Delimitations

Assumptions

Assumptions are the beliefs and opinions a researcher considers true and imposes on the study (Kirkwood & Price, 2013). Assumptions are a basic part of a research problem and shape the study undertaken by a researcher (Kirkwood & Price, 2013). Kirkwood and Price (2013) contended that researchers' beliefs influence their research and underexamined assumptions lead to questionable findings. I assumed the participants of the study understood my interview questions. I assumed the participants of the study gave honest responses to my interview questions knowing their responses would be private and confidential. I assumed the participants did not discuss any part of the interview process with other participants until all interviews were completed. I assumed the inclusion criteria of the sample were appropriate ensuring the participants were knowledgeable and experienced with software development, technical debt, and collaboration. I assumed the participants would give responses that were representative of my study population. I assumed the application of a qualitative approach for this case study would provide accurate data and constructs for exploration.

Limitations

Every study has limitations due to restrictions on the research question and the study's research methods (Denscombe, 2013). Denscombe (2013) defined limitations as restrictions on the interpretations and conclusions of a study due to the chosen research areas and methods. There may have been unknown circumstances or factors at the location where my participants work that could have biased their responses. The number of participants available may have been inadequate to reach saturation. The data collection consisted of interview questions and procedural documentation, which may have limited the findings. The study was limited to software development leaders employed by a large health care provider in the state of California, which may have limited the representability of the study. I limited interview participants to those who are actively involved in software development and excluded IT professionals outside of software development.

Delimitations

Delimitations are statements about items the researcher believes are outside the boundaries of the research problem (Denscombe, 2013). Denscombe (2013) contended that delimitations are the boundaries of research, and items outside these boundaries are not relevant to the research problem. A delimitation of this study was the inclusion of participants who actively worked in software development; I excluded other project stakeholders. A second delimitation was the interview questions, which were limited to software development strategies to minimize technical debt. The third delimitation was a single organization in the California. A fourth delimitation was health care providers. A

fifth delimitation was the relatively small sample size. A larger sample size would have been costlier and more time-consuming.

Significance of the Study

Contribution to Information Technology Practice

The significance of this study was to increase awareness of how collaboration among software development teams can affect the amount of technical debt accumulated by those teams. Collaboration in various forms is important to the success of IT projects, but this study provided insight into which methods, types, and characteristics of collaboration are important from a software development standpoint. I investigated whether the collaboration instrument or frequency of collaboration affected the accumulation of technical debt. I examined how the participants in the collaboration may drive the choice of instrument and frequency. I also explored the preferred collaboration strategies of software development leaders and the acceptance of these collaboration techniques by software developers.

This study was significant for IT executives, IT project managers, software development leaders, and software developers. The study provided IT executives with knowledge regarding the most effective means of collaboration among software development teams so leaders can guarantee their organizations have the knowledge, tools, and infrastructure to support these collaboration methods. This study benefited IT practitioners by identifying collaboration strategies that software development leaders can implement to minimize the technical debt accrued by their software development teams. The study benefited IT organizations by establishing collaboration best practices

that will help minimize technical debt and save organizations time and money. The study provided IT project managers with a better understanding of the collaboration strategies they should implement in their IT projects.

Implications for Social Change

This study effected social change by demonstrating how collaboration among individuals can be used to solve collective problems. Collaboration brings people of diverse backgrounds, different perspectives, and varying skill sets together to achieve a common goal. Collaboration, sharing problems, and working together extend far beyond the workplace into personal lives. The findings from this study may effect positive social change both inside and outside of the workplace.

A Review of the Professional and Academic Literature

The purpose of this qualitative case study was to explore collaboration strategies software development leaders use to reduce the amount of technical debt created by software developers. The focus of the literature review was the research question: What collaboration strategies do software development leaders use to minimize technical debt created by their software developers? I explored the TAM, technical debt, collaboration in software development, and collaboration strategies that software development teams use.

This literature review comprises 90 articles, journals, and conference proceedings. The primary research libraries and databases included the ACM Digital Library, EBSCOhost Computers and Applied Sciences Complete, IEEE Xplore Digital Library, ScienceDirect, ProQuest Computing, and ProQuest Dissertations and Theses Global. I

also used the Google Scholar search engine. I identified the peer review status of articles using Ulrich's Global Serials Directory. I reviewed 91 articles, of which 81 (89%) were peer reviewed and 77 (85%) were published within 5 years of my anticipated graduation date.

The literature focused on four key areas: (a) the TAM2, (b) technical debt in software development, (c) collaboration in software development, and (d) collaboration strategies. This review of the TAM focused on the perceived usefulness, perceived ease of use, cognitive influences, and social effects relating to collaboration and software development. The research into technical debt involved the history, causes, types, consequences, identification, and management of technical debt. The research into collaboration included benefits, strategies, technologies, effects on software quality, and overall use in software development.

Technology Acceptance Model

Davis (1986) introduced the TAM to explain users' attitude toward and behavioral intention to use a system (Figure 1). Davis hypothesized that a user's attitude toward and behavioral intention to use a system are major determinants influencing the user's actual use of the system. In this study, I explored the attitudes and intention of software developers to use collaboration techniques to reduce technical debt.

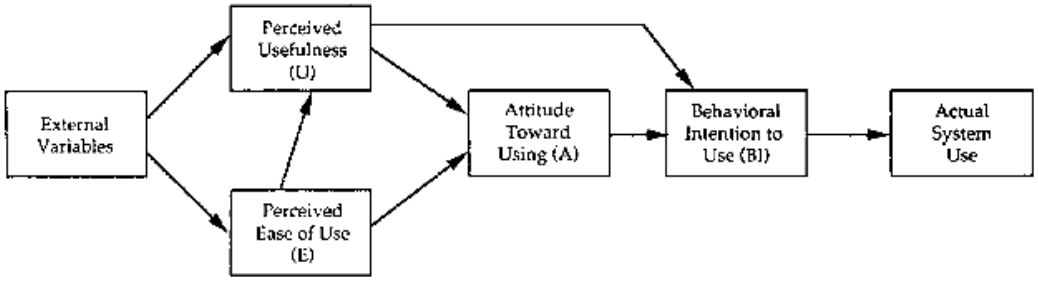

Figure 1. Technology acceptance model. Reprinted from "User Acceptance of Computer Technology: A Comparison of Two Theoretical Models," by F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, 1989, *Management Science, 35*(8), p. 985. Copyright 1989 by INFORMS. Reprinted with permission (Appendix C).

Davis (1986) developed the TAM as a variation of Fishbein and Ajzen's theory of reasoned action (TRA), a theory that was adapted for modeling user acceptance of information systems. Davis posited that the attitudes toward and behavioral intention to use a system are driven by the perceived usefulness and perceived ease of use of the system. Davis defined *perceived usefulness* as a user's subjective view that using a specific system will improve his or her performance within an organization. Perceived usefulness directly affects a user's attitude toward and behavioral intention to use a system. Davis defined *perceived ease of use* as the degree to which a person believes the use of a system will be free of effort. Perceived ease of use directly affects a user's attitude toward using a system. I investigated software developers' attitudes regarding the perceived usefulness of collaboration strategies to reduce technical debt and the ease of use to implement these collaboration strategies.

Qiu, Wang, and Yang (2015) posited that these two perceptions affect users' attitudes, positively or negatively, toward using a specific technology. Yucel and

Gulbahar (2013) juxtaposed perceived usefulness and perceived ease of use, explaining that they are predictors of users' acceptance of a technology. Abdullah and Ward (2016) maintained that users' attitudes toward a system influence their behavioral intention to use the system and ultimately determine the actual use of a system. Yucel and Gulbahar contended that perceived usefulness is the most important factor in determining behavioral intention to use a system. I explored whether the perceived usefulness and ease of use of collaboration strategies determined their actual use by software developers.

Davis et al. (1989) posited that the objective of TAM is to explain the factors of technological acceptance that are capable of theoretically justifying user behavior throughout a comprehensive range of technologies by discovering the influence of external factors on beliefs, attitudes, and intentions. Wallace and Sheetz (2014) asserted that the purpose of TAM is to explain why individuals choose to accept or reject a specific technology to complete a given task. I examined why software developers choose specific collaboration technology to reduce technical debt. Yucel and Gulbahar (2013) contended that TAM is applicable to a wide range of user populations to understand how users try new technologies, and that TAM could predict user acceptance of tools by determining the effect of modifications to those tools on user acceptance. I explored how modifications to these collaboration strategies may influence their use.

Social and cognitive influences. Venkatesh and Davis (2000) extended the original TAM by adding additional theoretical constructs spanning social influence processes and cognitive instrumental processes (Figure 2) to gain a better understanding of the determinants of perceived usefulness and usage intention. The main objective of

the extended model, or TAM2, was to determine the antecedents of these external influences that affect perceived usefulness (Yucel & Gulbahar, 2013). Venkatesh and Davis contended that the social influence processes in TAM2 include subjective norm, voluntariness, and imagination, whereas the cognitive instrumental processes include job relevance, output quality, and result demonstrability.

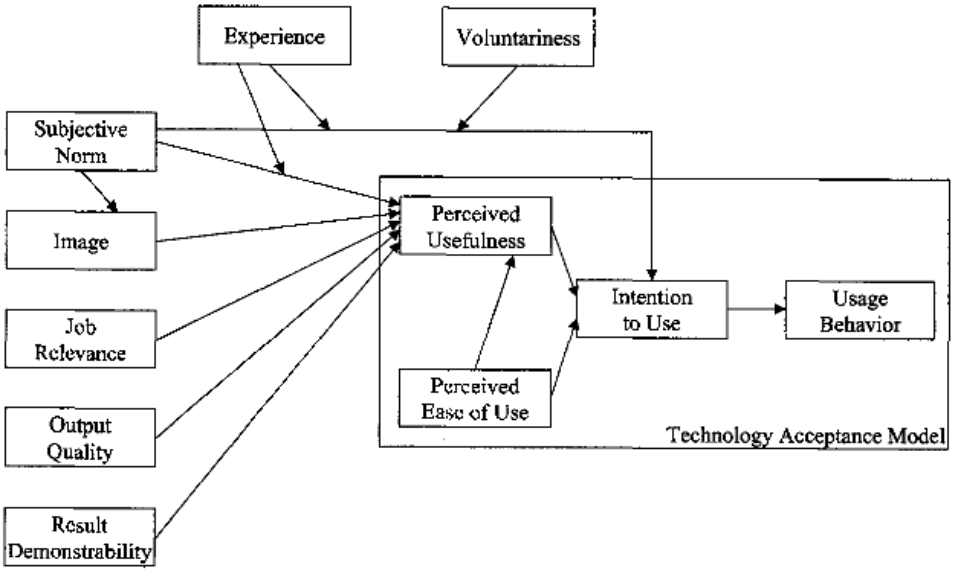

Figure 2. Extension of the technology acceptance model. Reprinted from "A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies," by V. Venkatesh and F. D. Davis, 2000, *Management Science, 46*(2), p. 188. Copyright 2000 by INFORMS. Reprinted with permission (Appendix C).

Riemenschneider, Hardgrave, and Davis (2002) defined *subjective norm* as the degree to which people think that others who are important to them believe they should perform a behavior and found it to be a significant determinant of perceived usefulness and intention for use. Martinez, Cachero, and Melia (2013) applied subjective norm to software development by defining it as the degree to which developers think that others who are important to them believe they should perform a specific behavior. Martinez et al. found that developers are more likely to use a method when they believe others who

are important to them consider that they should use it. I explored whether subjective norms influence software developers' attitudes toward using specific collaboration strategies.

Park, Rhoads, Hou, and Lee (2014) defined *voluntariness* as the extent to which a person believes the acceptance of technology is not mandatory. Riemenschneider et al. (2002) found that voluntariness significantly moderates the direct effect of subjective norm on intention to use. Martinez et al. (2013) found that adopting new development methods requires sizable mental efforts, and developers perceiving adoption as voluntary are less likely to adopt such methods. Venkatesh, Morris, Davis, and Davis (2003) defined *image* as the degree to which a person believes innovation use will enhance his or her image or status in a social system. The assumption is that people in an organization who use the innovation have more prestige and higher status than those who do not use the innovation.

Lala (2014) defined *job relevance* as an individual's belief that a technology is applicable to his or her work. Individuals are more willing to accept new technology if they perceived it as relevant to their job, and they are less likely to accept it when seen as irrelevant to their job. Venkatesh and Davis (2000) theorized that output quality depends on job relevance and is used to measure the degree to which a technology performs a task that is relevant to a person's job. Riemenschneider et al. (2002) examined the ability of software developers to communicate the advantages of using an application development methodology as a method to test result demonstrability. I investigated how software developers view collaboration strategies as relevant to their job and improve the quality

of their software. This is significant because Li, Avgeriou, and Liang (2015) found poor software quality to be one of the most common classifications of technical debt.

Analysis of related theories. The basic supposition of the TRA and TAM is that an individual's intention to perform a behavior is influenced by the evaluation of his or her beliefs and subjective norms (Priyanka & Kumar, 2013). TRA explores a wider range of behavioral beliefs whereas TAM narrows the focus of an individual's beliefs to perceived usefulness and perceived ease of use of technology. I focused on these two core beliefs, which made TAM more suitable for my study. TRA is limited to subjective norms, whereas TAM2 explores a wider range of social influences such as subjective norms, image, job relevance, and output quality, leading me to surmise that these social influences affect the belief of perceived usefulness. I explored the wider range of social influences present in TAM2.

The theory of planned behavior (TPB) is an extension of TRA that adds the construct of behavioral control beliefs, which are a person's perceptions of his or her ability to perform a behavior (Yucel & Gulbahar, 2013). This construct is similar to TAM's perceived ease of use, which is a person's belief that the use of a system will be free of effort. TPB does not extend the social influences of TRA, limiting it to subjective norms. My research required examination of a wider range of social influences such as job relevance and output quality, which are not present in TPB.

Khayati and Zouaoui (2013) posited that TAM's concept of perceived usefulness was based in part on the self-efficacy theory supposition that the expected results of a behavior influence the intention to use the behavior. Jun, Lee, and Jeon (2014) concluded

that self-efficacy has a direct effect on perceived usefulness but not perceived ease of use. TAM and TAM2 were more suitable for my study, which addressed both perceived usefulness and perceived ease of use.

Diffusion of innovation (DOI) theory is used to understand individuals' attitudes toward and willingness to adopt technology based on its communication within an organization through time (Conrad, 2013). DOI is like TAM in that both models are used to understand user acceptance and adoption of technology. DOI differs in that it addresses the rate of adoption over time. I explored collaboration strategies currently used by software development leaders and not the amount of time to adopt these strategies.

Limitations of TAM. Although TAM remains a popular model for analyzing information system acceptance, some researchers believe that TAM has questionable heuristic value, limited explanatory power, a sense of triviality, and no practical value (Priyanka & Kumar, 2013). Fletcher, Sarkani, and Mazzuchi (2014) posited that researchers have extended TAM to encompass nearly 30 additional factors to explain additional sources of variance. Priyanka and Kumar argued that the data collection approach for TAM is weak owing to its reliance on subjective, self-reported surveys rather than actual system use. Fletcher et al. contended that users might perceive usefulness in a system but reject the system owing to poor reliability or lack of user support mechanisms. Users reject systems that provide a significant amount of usefulness by manifesting negative attitudes regarding poor reliability and support mechanisms. Svendsen, Johnsen, Almas-Sorensen, and Vitterso (2013) found that certain personality traits such as extraversion and emotional stability could affect a person's perceived

usefulness and behavioral intent to use technology. These additional determinants lessen the accuracy and predictability of behavioral intent. Fletcher et al. argued that the timing and frequency of TAM data collection focus on the active decision-making process at a single point in time and not user adoption after the initial decision. The focus on a single point in time is an important aspect in that a person may initially accept a technology but reject the technology after a short period.

Usage of TAM in research. Researchers have applied TAM to a wide variety of industries and contexts. Rodrigues, Oliveira, and Costa (2016) employed TAM to assess the influence of determinants on the adoption of applications in the e-banking industry. Rana, Dwivedi, and Williams (2013) used constructs from TAM2 to explore the adoption of e-government services. Biederer, Arguel, Liu, and Lau (2014) studied user acceptance of mobile applications in the health care industry. The use of TAM in a broad and diverse collection of industries and organizations supports the use of TAM in software development organizations. Cheung and Vogel (2013) used TAM to explain factors that influence the acceptance of Google applications in a collaborative e-learning environment. Lee and Lehto (2013) used TAM to identify determinants affecting behavioral intention to use YouTube for procedural learning. Polancic, Jost, and Hericko (2015) explored TAM's effects on team-based e-collaboration at a Fortune 1000 company. The use of TAM in research involving collaboration, learning, and teams supported the use of TAM research in software development collaboration, which was the purpose of my study.

Research using TAM has expanded into software methodologies, agile processes, software quality, and other areas of software development practices beyond physical systems (Martinez et al., 2013; Overhage & Schlauderer, 2012; Wallace & Sheetz, 2014). Several studies have found TAM beneficial for studying software developers' intention to use software development methodologies (Martinez et al., 2013; Riemenschneider et al., 2002). Similar studies have explored TAM to identify factors influencing software developers' use of various software development practices (Overhage & Schlauderer, 2012; Vijayasarathy & Turk, 2012). Researchers have even used TAM to evaluate software measures (Nel, Nel, & Cronje, 2016) and software process improvement (Wallace & Sheetz, 2014).

Usage of TAM in collaboration. Several researchers have applied the TAM and its extension (TAM2) to study collaboration technologies. Di Russo and Douglas (2013) posited that perceived usefulness and perceived ease of use of document sharing technology is the best predictor for the adoption of the technology for collaboration purposes. Their study examined the satisfaction and adoption levels of SkyDrive, Facebook Docs, and Google Docs as collaboration tools. Google Docs was the most widely adopted document sharing technology and had the highest levels of perceived usefulness and perceived ease of use (di Russo & Douglas, 2013). Software developers may use document sharing as a strategy for collaborating on project requirements, product specifications, source code, procedures, processes, methodologies and project information. Godin and Goette (2013) applied constructs of the TAM2 to identify factors that contribute to the use of an on-demand collaboration, online meeting, web

conferencing and video conferencing application by Cisco Systems. Godin and Goette established that the performance expectancy and effort expectancy of the collaboration technology significant effects on the intention to use the collaboration tool. Their study measured perceived usefulness for communication, job relevance, and ease of use. Godin and Goette also theorized that social influences and subjective norm have significant effects on the intention to use collaboration tools. Godin and Goette expanded the usage of TAM constructs beyond asynchronous document sharing to synchronous video conferencing where social influences may be present. Video conferencing is an important aspect of software development organizations allowing simultaneous collaboration of team members regardless of geographic location or time constraints.

Huang, Hood, and Yoo (2013) employed constructs of the TAM to investigate collaborative learning tools used by university students. They specifically examined blogs, wikis, social networking communities, online video sharing, online games, and an immersive virtual environment. Huang et al. posited that the usefulness of the collaboration technology in learning a task and completing a task more efficiently has a positive effect on the intention to use collaboration technology. Huang et al. expanded the applicability of the TAM to Internet-based collaboration technologies used by software developers to collaborate with one another, share knowledge, and learn from each other. Cheung and Vogel (2013) applied constructs of the TAM to investigate the use of the Google applications suite of tools as a means of collaboration in project-based environments. The Google tools included email, chat, document sharing, video conferencing, and other collaboration tools.

Cheung and Vogel (2013) hypothesized that the perceived usefulness and ease of use of collaboration tools positively influence the attitudes towards and intent to use project-based collaboration tools. Project-based collaboration using a range of technology and tools is an important characteristic of software development projects (Cheung & Vogel, 2013). Polancic et al. (2015) applied the TAM2 to investigate the individual and collaborative work productivity using cloud-based modeling tools in software development. They found that job relevance, perceived usefulness, output quality, and image were deciding factors determining which modeling tools software developers choose. Their study is significant in that it established a relationship between collaboration tools and software development.

Usage of TAM in software development. Researchers studying software development processes, methods, and practices have used the TAM and TAM2 to gain critical knowledge. Riemenschneider et al. (2002) applied TAM2 to gain a better understanding of the determinants of software developers' intention to use an application development methodology. Riemenschneider et al. established that perceived usefulness, subjective norm, and voluntariness were substantial determinants in software developers' intention to use an application development methodology. The significance of this study is applying TAM2 to the acceptance of software development methodologies and not just technology.

Martinez et al. (2013) also extended the use of the TAM to explore whether software development methodologies chosen by software developers were dependent on the type of application developed. Martinez et al. hypothesized that software developers

choose the software development methodology they perceived as most useful and easiest to use regardless of the type of application they are developing. Qiu, Wang and Yang (2015) contend that the perceived usefulness and perceived ease of use of a secure software development methodology plays a significant role in the spreading and digestion of innovation among team members. A significant aspect of these studies is expanding the use of TAM to include software development methodologies, which plays a significant role in the accumulation of technical debt and can be a determining factor in the type of collaboration a software development project uses.

Vijayasarathy and Turk (2012) applied TAM and TAM2 to gain a better understanding of factors that influence software developers' adoption of agile processes and methods. Vijayasarathy and Turk theorized that perceived usefulness and subjective norm play a significant role in influencing software developers' adoption of agile processes and methods. This study extends the use of the TAM beyond software development methodologies to include software development processes as well. Overhage and Schlauderer (2012) applied TAM constructs to investigate the long-term acceptance of agile methodologies by software developers. Overhage and Schlauderer found the perceived usefulness of the agile scrum process by software developers and the relevance to their job were driving factors in the long-term acceptance of agile methodologies by software developers. These two studies highlight the importance of TAM to predict the initial adoption and long-term acceptance of agile processes and methods, which go beyond IT teams and can encompass entire organizations.

Wallace and Sheetz (2014) extended TAM to study the perceived usefulness of software measures in project management and software process improvement. Wallace and Sheetz posited that software developers are more likely to adopt software measures perceived as useful and easy to use in their software development practices. Nel et al. (2016) applied TAM to gain a better understanding of the use of quality appraisal techniques and process measures to improve software development practices. Nel et al. contend usefulness, ease of use, result demonstrability, subjective norm and career consequences are factors that influence the usage of quality appraisal techniques to improve software development practices. The importance of these studies is applying the TAM to software and quality measures in software development.

Researchers are applying the constructs of TAM to study technical debt in software development organizations. Holvitie, Leppanen and Hyrynsalmi (2014) explored the perceived usefulness of technical debt knowledge on agile software development processes and practices. The study found the perceived usefulness of certain collaboration strategies has a positive effect on a project's technical debt. Eliasson, Martini, Kaufmann, and Odeh (2015) investigated the perceived usefulness of metrics chosen to identify and measure architectural technical debt on the ability to communicate and share technical debt knowledge with others. Eliasson et al. found the perceived usefulness of technical debt metrics influences the success of technical debt collaboration with others. Li, Liang, and Avgeriou (2015) explored the perceived usefulness and perceived ease of use of software development approaches to identify technical debt.

They found that both perceived usefulness and ease of use influence software developers' intention to use a specific approach to identify technical debt.

Usage of TAM in technical debt. Several studies on technical debt have explored some of the constructs of the TAM. Li, Liang, and Avgeriou (2015) explored the perceived usefulness and perceived ease of use of approaches for identifying architectural technical debt. Eliasson et al. (2015) found the perceived usefulness of the metrics chosen to identify and measure technical debt may affect the ability to understand and communicate the debt to others. The TAM constructs perceived usefulness and perceived ease of use have implications for practice by aiding in the determination of which identification, measurement, and management approaches an organization should implement.

Technical Debt

Fagan (1976) was the first researcher to propose that program design and coding errors eventually require corrections at some point in time. Fagan theorized that these corrections are costlier when performed later in the software development process. Errors that go undetected during the software development process or those software developers ignore will incur a future and costlier responsibility to correct. Ward Cunningham (1992) first published the concept of technical debt in a report at OOPSLA'92 where he proposed the concept of technical debt as shipping software with immature architecture and "not quite right" code incurring future development costs. Cunningham compared technical debt to financial debt where a tradeoff to save time or money upfront eventually requires repayment of the debt with interest. Yli-Huumo, Maglyas, and Smolander (2016)

contend that technical debt often arises from the conflict between software engineering best practices and business decisions. Poor software engineering practices may incur undetected debt while business decisions may incur technical debt that developers ignore.

Ampatzoglou et al. (2015) defined technical debt as the increase in costs associated with maintaining or enhancing a system resulting from convenient shortcuts taken during the development of the system that ignores industry best practices. Holvitie and Leppanen (2015) suggested a software project might incur technical debt by not following best practices in regards to process, testing, architecture, implementation, and documentation. Tom et al. (2013) posited that technical debt is the consequence of poor software development practices and accumulates when software engineers do not follow industry best practices. A software development project accumulates technical debt when the team does not follow a software methodology or does not adhere to the principles of object-oriented design and programming.

In contrast, Congyingzi and Yan (2016) more broadly defined technical debt as decisions made today to reach short-term goals at the expense of creating future work. Li, Avgeriou, and Liang (2015) suggested that technical debt may occur outside of software development and is the abandonment of recognized best practices in the areas of organizational management, project management or engineering that negatively affect time, resources or cost. Technical debt may originate from any person or area in an organization but culminates with software development organizations repaying the debt. Tom et al. (2013) found code decay, deteriorating architecture, insufficient documentation, inadequate testing, and missing requirements as contributing factors to

technical debt. The decision of agile software development teams to focus on being more responsive to customer needs and delivering software as quickly as possible often outweighs the use of best practices thus creating technical debt (Martini, Bosch, & Chaudron, 2015). Factors affecting the accumulation of technical debt may exist outside of the software development environment.

The most recent definition of technical debt advanced by Foganholi et al. (2015) states that technical debt refers to the long-term costs resulting from the postponement of code optimizations, defect corrections, documentation or any other best practice during software development to meet time or financial constraints. Foganholi et al. further contend that postponing any technical activity during software development for any reason results in the accumulation of technical debt. A decision to postpone a technical activity and incur technical debt may originate from any stakeholder on a project or in the organization. A decision suggests an intentional act such as deciding to forego best practices or deciding to use an imperfect design to meet a deadline.

Technical debt in software development. Curtis, Sappidi, and Szynkarski (2012) found that technical debt has adverse effects on software quality by decreasing the reusability, flexibility, understandability, effectiveness, functionality or the extendibility of the software. Femmer, Fernandez, Wagner, and Eder (2016) found ambiguities and incomplete software requirements may create time delays, cost overruns and negatively affect software quality in software development projects leading to technical debt. Ramasubbu and Kemerer (2015) found technical debt might cause unpredictable ripple effects and propagate errors within an enterprise software system due to the myriad of

interconnections and interdependencies between various modules making it difficult to assess the impact of technical debt on system reliability and estimate the time and effort required to make software changes. Ramasubbu, Kemerer, and Woodard (2015) of enterprise systems at 48 Fortune 500 companies found that technical debt increases the chance of system failures by up to 62% and causes a three-fold increase in the error backlog. Curtis et al. (2012) estimated that every line of code in a software application has $3.61 of technical debt. Based on these researchers, technical debt is present in many organizations resulting in severe consequences to software quality thus increasing the volatility of systems and costs associated with maintaining those systems.

Ramasubbu and Kemerer (2014) purported that incurring technical debt does improve customer satisfaction in the short-term by speeding up delivery times and quickly adding new functionality. In contrast, Ramasubbu, Kemerer, and Woodard (2015) found technical debt decreases long-term customer satisfaction by 50% and software teams delivering software later by avoiding technical debt had 13 times fewer unresolved errors and seven times lower costs for bug fixing allowing them to deliver higher quality software with lower maintenance costs. Avgeriou, Kruchten, Nord, Ozkaya, and Seaman (2016) contend technical debt affects the longevity of information systems and their ability to evolve with changing conditions. Technical debt may have some short-term advantages, but it has significant long-term disadvantages in the areas of cost, quality, information system life expectancy and customer satisfaction.

Li, Avgeriou, and Liang (2015) postulated that technical debt could originate from the requirements, design, code, test, build, documentation, implementation or

maintenance phases of software development projects and it may originate from any stakeholder on a project. Many of the software development phases outlined by Li, Avgeriou, and Liang often involve team member collaboration and knowledge sharing. In contrast, Avgeriou et al. (2016) contend technical debt pervasively affects all aspects of software development including the management of the software development organization and team member collaboration. This study implies collaboration strategies that software managers implement may affect technical debt. Curtis et al. (2012) found undocumented methods was the top coding violation contributing to technical debt. Documenting code is one method team members use to collaborate and share knowledge in many of these software development phases.

Tom et al. (2013) categorized various types of technical debt including code debt, design debt, and architectural debt. Motherudin and Moksen (2015) posited that failure to follow software development best practices leads to increased technical debt. Zazworka et al. (2014) identified code debt, design debt, and architectural debt resulting from failure to follow object-oriented best practices. This is important as software development best practices that may increase technical debt if not followed correctly often include collaboration among team members and the use of collaboration tools by team members.

Alves et al. (2016) define code debt as source code problems related to bad coding practices that negatively affecting code legibility making maintenance more difficult. Krishna and Basu (2015) found inadequate coding standards, compromising software development processes, lack of code reviews and unit tests may result in the accumulation of code debt. Curtis et al. (2012) found poorly optimized algorithms,

unstructured code, duplicated code, inadequate comments, overly complex code, and other coding violations as contributing factors to code debt.

Tom et al. (2013) define design debt as software design with an inadequate focus on maintainability and adaptability, or a piecemeal design lacking sufficient refactoring. They contend architectural debt results from poor upfront solutions or solutions that become inadequate over time. MacCormack and Sturtevant (2016) found that deficient software architecture creates additional software maintenance efforts in the future incurring extra maintenance costs. Martini et al. (2015) found a lack of software developer knowledge due to inexperience, lack of domain knowledge, ignorance and carelessness could accumulate architectural technical debt. Lack of collaboration and knowledge sharing among software developers might lead to the accumulation of technical debt. Tang and Lau (2014) described a relationship that exists between design and architectural decisions where changes in one might affect the other. This indicates that design debt may contribute to architectural debt.

Tom et al. (2013) found that poor communication and collaboration processes make it easier for technical debt to accumulate without detection and this lack of awareness makes it easier for individuals to make additional decisions incurring more debt. Li, Avgeriou, and Liang (2015) found communication an important factor in making architectural debt visible to other stakeholders fostering discussions on managing the debt. Li, Avgeriou, and Liang further found certain communication approaches such as dashboards and lists of the debt allowed all stakeholders to have knowledge of the debt in a project. Codabux and Williams (2013) found communication and collaboration

among software teams promote a culture of knowledge sharing between the teams that may help reduce architectural debt. These researchers support that poor collaboration among software development teams may exacerbate factors influencing the accumulation of technical debt whereas healthier collaboration may reduce these effects.

Technical debt management. Guo, Spinola, and Seaman (2014) posited that technical debt management is a repetitive process of identifying a list of items containing technical debt, measuring the amount of technical debt, deciding which items to repay and determining when to repay them. In contrast, Li, Avgeriou, and Liang (2015) argue that the purpose of measuring technical debt is to quantify both the cost and benefit of repaying the debt. They further contend that management of technical debt should include prioritization and prevention activities. Avgeriou et al. (2016) postulate that economic investment theories are good techniques for prioritizing technical debt. A review of 100 research studies by Alves et al. (2016) found the cost-benefit analysis, portfolio approach, and options as the three most common strategies for managing technical debt. The purpose of identifying and measuring technical debt is to facilitate the decision-making process to prioritize repayment of the debt and future prevention strategies. The cost-benefit analysis is the most common strategy for managing technical debt repaying the items with the highest payoff first (Avgeriou et al., 2016).

The identification of technical debt is the first and most important process in managing technical debt (Avgeriou et al., 2016). Li, Avgeriou, and Liang (2015) found the most common artifact in identifying technical debt is source code and most common approach is code analysis. A review of 100 studies by Alves et al. (2016) found the most

frequent technical debt identification techniques were code smells, automatic static analysis issues, modularity violations and structural issues. Zazworka et al. (2014) theorized that these four approaches have very little overlap thus characterizing different problems with source code. Li, Avgeriou, and Liang (2015) found the most common classifications of technical debt were coding violations, duplicate code, code smells, complex code, structural issues, and low-quality code.

Curtis et al. (2012) advanced a technique for measuring technical debt by analyzing the number of should-fix violations, the hours required to fix them and the labor cost. In contrast, Izurieta and Bieman (2013) measured technical debt by studying the aging of design patterns and the extent to which designs decay, rot, and accumulate grime. Guo et al. (2014) contend that estimating the cost of overhauling software to realize the ideal level of software quality was one approach to quantifying technical debt. Measuring technical debt is dependent on the identification process and is a prerequisite for the decision-making process of prioritizing, repaying, and preventing debt.

Collaboration in Software Development

Holvitie et al. (2014) found the perceived usefulness of agile software development practices have a positive effect on reducing and preventing technical debt. They also found the perceived usefulness of collaborative processes such pair programming, peer reviews, and retrospectives as positive or very positive. Tom et al. (2013) contend that poor collaboration decreases the visibility of technical debt making it easier to accumulate and easier for developers to take shortcuts increasing the technical debt. Guo et al. (2014) posited that collaboration in code reviews is a method developers

use to identify violations of best practices incurring technical debt. These researchers have identified that most of the processes regarding technical debt identification and management may benefit from knowledge sharing, communication, and collaboration.

Magdaleno, de Oliveira Barros, Werner, de Araujo, and Batista (2015) contend collaboration reduces the time required to solve problems, improves problem-solving, generates creative alternatives, enhances decision-making, fosters learning, encourages innovation, and improves job satisfaction. Inayat and Salim (2015) found software engineers use collaboration to facilitate defect discussions, coding issues, code reviews, refactoring, software quality improvements, software maintenance, software design, requirements analysis, and planning activities. Software development is a complex process requiring cooperation, communication, and collaboration with software engineers, architects, designers, database administrators, and other project stakeholders. Ferzund, Yasrab, and Razzaq (2014) contend that software engineers spend 70% of their time working or collaborating with others. Ferzund et al. argued that software developers could spend more than 70% of their time collaborating with others in large projects. Giuffrida and Dittrich (2015) contend collaboration is a basic component of software development where teams use version control, bug tracking, email, web pages, instant chat, code reviews, and documentation to coordinate activities and distribute knowledge. Software developers devote a lot of time to collaboration using a wide range of tools and technology to facilitate their collaboration efforts.

Magdaleno et al. (2015) define collaboration as people with complementary skills coming together to solve complex problems which none of them could do individually.

Inayat and Salim (2015) juxtaposed collaboration is the exchanging information among team members and sharing awareness knowledge of others. There are two important concepts in the definition provided by Inayat and Salim. The first is the lack of a predefined purpose for collaboration. Unlike Magdaleno et al. who contend the purpose of collaboration is to solve complex problems, Inayat and Salim contend collaboration may have many purposes. The second important concept is that collaboration involves the sharing of awareness knowledge. Molina, Gallardo, Redondo, and Bravo (2015) define awareness as understanding who is working with you, what they are doing and how your own actions interact with theirs. Molina et al. contend awareness is useful for coordinating actions with others, discussing tasks, anticipating the actions of others and finding others to help. Magdaleno et al. contend collaboration consists of communication, coordination, group memory, and awareness. The collaboration consists of exchanging information, organizing work, operating in collective environments, and being cognizant of team member activities. I define collaboration in terms of software development as software engineers sharing ideas, concepts, knowledge, and awareness to improve software quality.

Benefits of collaboration. Ferzund et al. (2014) found that better collaboration and communication increases the interaction among software developers and significantly improves software quality. Ferzund et al. found communication and collaboration enhance software development processes. In contrast, Caglayan and Bener (2016) found that developer code changes with many direct collaborators introduce a higher number of defects in the code. This may indicate that developers with many

collaborators might be making changes to areas of the code others are also changing increasing the chance of conflicting changes. Caglayan and Bener indicate the benefits from the number of collaborators may peak at some point, after which more collaborators might become a burden to the software developers.

Mangalaraj, Nerur, Mahapatra, and Price (2014) found collaborating pairs have higher job satisfaction and deliver better quality software designs than many individual developers. Ramasubbu, Kemerer, and Hong (2012) found the perceived usefulness of collaborative programming strategies on maintenance tasks was higher than that of independent programming strategies. Di Bella et al. (2015) observed collaborating pairs have lower defect rates and introduce fewer new defects than individuals performing similar tasks. Di Bella et al. contend the amount of collaboration among pairs significantly influences defects rates by enhancing developers' knowledge over the code. Mangalaraj et al. posited that pairs often require more time to complete tasks than individuals assigned similar tasks. Although software developers collaborating in pairs require more initial time to complete tasks, they might save time in the end by improving software quality and introducing fewer defects.

Milovanovic, Minovic, Stavljanin, Savkovic, and Starcevic (2012) contend software developers use collaboration extensively in the analysis, design, debugging, and support phases of software development to increase task performance. Milovanovic et al. (2012) found that collaboration and knowledge sharing improves development processes and increases the problem-solving ability of software development organizations. Licorish and MacDonell (2014) posited that software developers' level of knowledge

sharing is positively correlating to their task performance. Ozer and Vogel (2015) posited that software development organizations perform better when adopting formal, rather than informal, processes for sharing knowledge. Software development organizations that establish formal collaboration and knowledge sharing practices will improve development processes and performance in all phases of software development.

Xiang, Lu, and Gupta (2013) hypothesized that knowledge sharing is positively correlated with software development team performance and increasing the shared mental model of teams leads to better team performance. Tang (2015) contends timely, accurate information exchange improves the performance of software development teams by cultivating a shared understanding of each other's areas of knowledge. Ferzund et al. (2014) found greater team member interaction from collaboration activities improves software development performance. Software developers who better understand their teammates' activities and capabilities are better able to exchange help will other developers increasing the performance of the team.

Ramasubbu et al. (2012) purported that teams employing collaborative programming strategies were more productive than teams using independent programming strategies. Ramasubbu et al. found the perceived ease of use of collaborative programming was 28% higher than that of independent programming. Licorish and MacDonell (2014) posited that collaboration and knowledge sharing among software developers benefits weaker team members. Mangalaraj et al. (2014) found software development organizations could improve the performance of weaker software developers by pairing them with other developers to collaborate and share knowledge on

tasks. Ozer and Vogel (2015) found software developer performance increased when they received knowledge from other software developers in the organization. Software development organizations utilize collaboration and knowledge sharing to improve the knowledge, skills, and performance of inexperienced team members.

Ferzund et al. (2014) contend the availability of better forms of collaboration improves the bug fixing processes, which enhance software quality. Ferzund et al. found web-based communication and collaboration processes leads to higher rates of bug fixing. Di Bella et al. (2015) found developers using pair programming during defect correction activities helps reduce the introduction of new defects because of the defect correction process. Ramasubbu et al. (2012) found software development organizations could reduce maintenance efforts by up to 70% by assigning tasks that are more complex to collaborative programming teams. There are many forms of collaboration software organization may incorporate into their defect correction practices. Certain forms of collaboration are better suited for completing complex tasks.

Collaboration Strategies

Guo, Spinola, and Seaman (2014) posited that a code review is one method developers use to identify violations of best practices incurring technical debt. McIntosh, Kamei, et al. (2015) define a code review as the collaboration between the authors of programming code and those reviewing the code. Foganholi et al. (2015) contend code reviews facilitate identification of violations of software development best practices. Software developers use code reviews to collaborate with one another to identify violations of best practices to prevent the accumulation of technical debt.

Fagan (1976) introduced the concept of design and code inspections based on formal meetings with heavyweight processes. Linhares, Borges, and Antunes (2012) purported software review meetings follow formal procedures involving groups of experts with designated roles. Linhares et al. postulated the purpose of these meetings is to discover discrepancies in software specifications, standards, and best practices to improve software quality. In contrast, McIntosh, Kamei et al. (2015) found the modern code review process uses a variety of web 2.0 technologies focusing on collaborative problem solving to improve software quality. The modern code review is a lightweight design and code inspection process with the same purpose as the formal meetings to improve software quality but in a shorter amount of time. McIntosh, Kamei et al. contend the modern code review process is more collaborative in nature focusing on code coverage, developer participation, and developer expertise.

Baysal, Kononenko, Holmes, and Godfrey (2015) found the most influential factor in the code review process is the level of participation by software developers. The more active developers are in the code review process the shorter the review time and the more likely others will accept their contributions. McIntosh, Kamei et al. (2015) found that lack of participation in code reviews has a negative impact on software quality and insufficient collaboration during code reviews correlates to higher defect rates in the code. Linhares et al. (2012) contend Agile and open source software development approaches emphasize collaboration and participation in software review meetings as a means of improving software development performance. Carver, Caglayan, Habayeb, Penzenstadler, and Yamashita (2015) found the number of collaborators during a code

review positively correlates to software quality. McIntosh, Kamei, Adams, and Hassan (2014) found the lower the number of reviewers and the lower proportion of changes that have been reviewed the more likelihood of an increase in postrelease defects. The number of participants, the amount of participation, and level of collaboration during code review processes have a direct impact on software quality and the performance of software development teams.

Linhares et al. (2012) contend software review meetings may involve developers, designers, and testers participating in quality assurance activities at various times during the development life cycle to verify the software has met the customer's requirements. McIntosh, Kamei et al. (2015) found that participation in code reviews has a larger impact on software quality than developer expertise. Caglayan and Bener (2016) contend developers with higher experience levels do not necessarily have lower defect rates. Caglayan and Bener found new developers tend to have their code reviewed more extensively producing higher quality software with lower defect rates than code from developers with greater experience with fewer code reviews. The number of code reviews and the participation rates by developers has a larger impact on software quality and future maintenance costs than the experience levels of the developers creating the code.

Carver et al. (2015) found more than 50% of code review comments focus on improving future maintainability and reducing future maintenance costs rather than identifying functional problems. McIntosh, Kamei et al. (2015) theorized that up to 75% of the issues resolved during code reviews point to significant issues that may result in defects later. Resolving these non-functional issues improves future maintainability of the

code. In contrast, Baysal et al. (2015) postulate that software development teams use code reviews to evaluate and improve the quality of code changes before committing the changes to the project repository. Code reviews are versatile processes to evaluate, identify, and improve code changes to enhance software quality, improve maintainability, and reduce maintenance costs.

Bacchelli and Bird (2013) theorized code reviews help teams share knowledge, improve awareness, create alternative solutions, foster transparency, and promotes shared code ownership among team members. McIntosh, Kamei et al. (2015) contend an important motivation for modern code reviews is the sharing of knowledge among software team members. Tang and Lau (2014) posited software architects use code reviews to share knowledge relating to design decisions and reasoning enabling reviewers to identify design issues and reflect on the acceptability of design solutions. Software solutions often lack sufficient documentation, communication, and design reasoning. Code and design reviews can facilitate creative solutions for solving design and software issues.

Pair programming. Jayalakshmi, Kavitha, and Niroza (2016) define pair programming as a software development technique where two programmers work together at the same computer to complete a single task. Di Bella et al. (2015) found that software developers working in pairs decrease the introduction of new defects and even low levels of pair programming during defect correction could significantly lower the overall defect density. Gupta, Bhattacharya, and Singha (2013) contend pair programming is a best practice of agile software development with extensive

collaboration between software developers who often switch roles to review one another's work. This type of pair programming is a formal process requiring one developer to write code while the other checks for errors, thinks of alternative solutions, and shares knowledge.

In contrast, Coman, Robillard, Sillitti, and Succi (2014) contend that pairs of software developers work informally to complete a task as needed. Coman et al. found that software developers spend 40% of their time employing informal pair programming as a means of overcoming unexpected technical difficulties completing a task. Coman et al. contend most software developer interactions are backup behaviors where team members come together to help each other as needed to complete tasks. These backup behaviors may include sharing knowledge, brainstorming, mentoring, and helping team members complete a task. Informal pair programming essentially refers to ad-hoc backup behaviors where developers collaborate to complete a specific task, including formal pair programming. Informal pair programming is a voluntary activity.

Jayalakshmi et al. (2016) posited that pair programming promotes quality programming skills, responsibility, mentoring, teamwork, and increased enjoyment among developers of varying experience levels. Gupta, Bhattacharya, and Singha (2013) contend software developers collaborate almost every minute during pair programming sessions covering each other's weaknesses. Breed, Mentz, and van der Westhuizen (2014) found improvements in the areas of planning, information management, monitoring and evaluation for individuals implementing pair programming. Plonka, Sharp, van der Linden, and Dittrich (2015) found programming pairs use strategies such

as indirect hints, pointing out problems, gradually adding information, giving clear instructions, providing explanations, and verbalization to exchange knowledge. These strategies are forms of collaboration software developers adapt to their needs and adjust their engagement level depending on the type and amount of knowledge they exchange on a given task.

Jayalakshmi et al. (2016) found developers using pair programming produce fewer defects, higher quality code and increase their knowledge of the system. Mangalaraj et al. (2014) contend developers working in pairs create higher quality designs and experience higher task satisfaction than many developers working independently. Di Bella et al. (2015) juxtaposed that while even low levels of pair programming may significantly lower the defect density of code, the benefits of pair programming on defect density peaks when developers spend over 30% of their time in pair programming. Paired development activities may offset lower skill levels of less competent team members, build teamwork, and improve software quality, but may incur higher development costs and reach a point where the costs outweigh the benefits.

Web 2.0 tools and technologies. Evans, Gao, Martin, and Simmonds (2015) posited that organizations are relying more on web-based tools, and technologies to communicate and collaborate. Tools such as email, chat, blogs, and wikis allow teams to create, organize, share, and critique knowledge with one another. Eservel (2014) found that software development organizations frequently use mailing lists, bug tracking, source control, and wikis for communication, collaboration, and knowledge creation among team members. Ferzund et al. (2014) found that software development organizations

using web-based tools for collaboration show significant improvements in development processes and software quality. The amount, variety, and effectiveness of collaboration tools available to software development teams affect interactions, relationships, and processes of software development teams leading to better software quality.

Giuffrida and Dittrich (2013) define a wiki as a website allowing users to share and management knowledge, coordinate activities, and improve awareness among teams by facilitating communication and collaboration. Menolli, Cunha, Reinehr, and Malucelli (2015) found that 54% of software development companies use wikis to create, store, and share knowledge and that wikis are the most widely used tool for this purpose. Milovanovic et al. (2012) found software developers consider the perceived usefulness of a wiki as improving software development processes, collaboration efficiency, and knowledge reusability. Wikis facilitate vital communication, interaction, and collaboration activities for software development organizations enabling them to improve their processes, efficiency, and software quality. Menolli et al. found software development companies perceive wikis as a usefulness and easy to use tool to share knowledge among software developers.

Esichaikul, Win, Bechter, and Rehman (2013) theorized that wiki collaboration improves communication, awareness, motivation, and differentiation of work. Cubric (2013) contends the use of wikis increases the knowledge of software developers and more importantly improves the quality of cognitive processes. Lavhengwa, van der Walt, and Lavhengwa (2014) found that wikis and other forms of e-collaboration improve knowledge development, foster innovation, and increases interaction with others.

Kasemvilas and Olfman (2013) developed a set of wiki extensions providing better support for synchronous and asynchronous communication among users, simplifying knowledge creation and improving the awareness of team member activities. Kasemvilas and Olfman found the extensions significantly increase the perceived usefulness and perceived ease of use of the wikis to improve awareness and collaboration of a team. Foganholi et al. (2015) found sharing of technical debt information was a critical aspect of successfully managing the reduction of technical debt.

Transition and Summary

This section contained an introduction to the problem of technical debt and collaboration in software development teams. The purpose of this study was to explore collaboration strategies software development leaders use to reduce technical debt. A qualitative case study approach was the most appropriate to answer the overarching research question to determine the collaboration strategies software development leaders use to minimize technical debt. The TAM2 provided the conceptual framework for viewing the perceived usefulness, perceived ease of use, and social influences of collaboration for reducing technical debt. The literature focused on the areas of technical debt in software development, software development processes, collaboration in software development and the TAM2.

Section 2 provides furthers details and justifications on the research methodology selected for this study. This section expands on the role of the researcher, sets the participant criteria, compares the research methodologies, and explores the population

sampling, ethical research, data collection, analysis, reliability, and validity. Section 3 will contain the results of the study based on an analysis of the data collected.

Section 2: The Project

This section provides additional details on the research method, design, and processes involved in this study. I defines the role of the researcher, criteria for participant selection, population sampling, and ethical research. I also explain the data collection, organization, and analysis processes and describe the issues of reliability and validity in the context of the study.

Purpose Statement

The purpose of this qualitative case study was to explore collaboration strategies software development leaders use to reduce the amount of technical debt created by software developers. The population for this study included senior software development leaders from a large health care provider in the state of California. The population was experienced with the phenomenon of technical debt and were involved in software development collaboration. These senior software development leaders participated in semistructured interviews to identify how collaboration of software development teams affects technical debt. The implications for positive social change include the potential to increase the reliability of communal software systems and reduce the economic burden of software on society by improving the collaboration among software development professionals.

Role of the Researcher

A qualitative researcher serves as the primary instrument of data collection and analysis in qualitative research (Lincoln & Guba, 1985; Sanjari, Bahramnezhad, Fomani, Shoghi, & Cheraghi, 2014). My role in this qualitative study was to collect, organize,

interpret, and analyze data in an unbiased manner. The role of the researcher includes investigating varying points of view and perspectives during the data collection process (Kavoura & Bitsani, 2014). A researcher develops good interview questions, listens to the participants' responses, does not have any preconceived notions or biases, understands the study issues, and is flexible during the interview process (Roulston & Shelton, 2015).

Transparent and methodical research helps preserve the integrity of a study, avoid personal biases, and avert distortions (Cronin, 2014). I have 18 years of professional experience in the field of software development working as a software engineer and software development leader. My personal experience with the study topic was my rationale for conducting this study. I did not have any personal or professional relationships with the participants.

I reviewed the Belmont Report (U.S. Department of Health & Human Services, 1979) regarding the ethical principles and guidelines for protecting participants in research studies. The Belmont Report identified respect for persons, beneficence, nonmaleficence, and justice as the key principles essential to ethical research (Hammer, 2016). A summary of the principles outlined in the Belmont Report helps participants avoid coercion and improper influences and the participants understand any risks involved in the study (Largent, Grady, Miller, & Wertheimer, 2013). I completed the Protecting Human Research Participants training offered by the National Institutes of Health (NIH) Office of Extramural Research (Certification Number: 1653141, Appendix A). I followed the principles set forth in the Belmont Report.

The possibility of bias exists in all research (Roulston & Shelton, 2015). The manipulation or distortion of data in qualitative research threatens the credibility of a study and may be unintentional or hidden from the researcher (Roulston & Shelton, 2015). Qualitative research requires objectivity to understand and interpret data (Reybold, Lammert, & Stribling, 2013). I approached this study as an independent observer gathering data and not inserting my personal beliefs into the study. I avoided bias by being aware of my own values, predispositions, and subjectivity in this study.

Researchers avoid bias by asking open-ended, nonleading questions to avoid influencing interview participants (Onwuegbuzie & Byers, 2014). I asked open-ended, nonleading questions during the interview process. I did not express my experiences or perceptions of the study topic to avoid influencing the participants and to avoid bias in the data collected. Recording the researcher's thoughts before and after interviews allows the researcher to identify and reflect on any potential biases that may have influenced a participant's response and skewed the data (Hoare & Hoe, 2013). I used a journaling approach to record my thoughts before and after each interview to prevent bias in my data.

The purpose of my interviews was to explore the participants' knowledge, experience, and perspectives on my research topic. An interview protocol provides a procedural guide for conducting interviews that includes preinterview scripts, postinterview scripts, interview questions, and interview reminders (Jacob & Furgerson, 2012). Researchers can mitigate potential ethical challenges between researchers and participants in qualitative studies by following research protocols (Sanjari et al., 2014).

An interview protocol helps protect participants in a study by ensuring participants' personal details are kept private (Qu & Dumay, 2011). I used an interview protocol (see Appendix B) that consisted of having participants sign a statement of informed consent, explaining confidentiality to participants, building rapport with participants, providing reminders to myself during the interviews, and explaining expectations after the interviews. Interview protocols help researchers develop broad, open-ended interview questions and help develop prompts to respond to any surprises to keep the interview on track and allow for any unexpected data (Jacob & Furgerson, 2012).

Participants

Defining participant criteria and selecting the most appropriate participants are fundamental aspects of a qualitative research study (DeFeo, 2013). The population and phenomenon of a research question are main factors in developing the inclusion criteria for participants in a study (Stern, Jordan, & McArthur, 2014). I clearly defined eligibility criteria aligned with my research questions and the phenomenon of my study. I only selected participants from a population that met my criteria to ensure the participants selected were appropriate for my study.

This usefulness of participants to a study depends on their understanding of the research phenomenon (Reybold et al., 2013). The objective of this study was to explore collaboration strategies used by software development leaders to reduce technical debt. I excluded participants without software development experience, not actively using collaboration, and lacking knowledge of technical debt. These participants will constrain any potential conclusion of the study. Potential harm to participants may result from

deselecting participants from a study (DeFeo, 2013). I avoided participant deselection harm by ensuring that I only selected participants with knowledge of my research phenomenon.

Determining the eligibility criteria of qualitative studies is a subjective activity requiring researchers to interpret and choose which participants are necessary for the study (Reybold et al., 2013). The participants' job title within their organization indicated they were a leader or held a senior level position on a software development team within the organization. The participants had at least 10 years of software development experience. The participants had experience with the phenomenon of technical debt and actively participating in collaboration within their software development team. The participants were actively using one or more of the following collaboration activities: design reviews, code reviews, project reviews, project retrospectives, pair programming, mentoring, team meetings, wikis or knowledge sharing. The participants had experience with using collaboration strategies to reduce technical debt. The participants were members of one of the two software development teams within the single line of business.

Strategies for gaining access to participants may involve sending potential participants a brief introduction, potential study benefits, study confidentiality, and convenience of the interview process (Hoyland, Hollund, & Olsen, 2015). Peticca-Harris, deGama, and Elias (2016) developed a dynamic, nonlinear process for gaining access to participants that includes four parts: study design and planning, identifying informants, contacting informants, and interacting with informants during data collection. My

strategy for gaining access to participants included planning, identifying participants, contacting participants, and interacting with participants. An important step in gaining access is to obtain approval from key stakeholders during the study design and planning stage (Peticca-Harris et al., 2016). I ensured I had approval from my committee members and permission from the Walden University Institutional Review Board (IRB) prior to conducting my study. Walden University's IRB approval number for this study is 03-24-17-0489027.

Gatekeepers are individuals who can facilitate access to participants by endorsing a researcher's work (Crowhurst, 2013). Gatekeepers recognize the value of a study, provide suggestions on gaining access, and have influence within an organization (Hoyland et al., 2015). I established trust and rapport with one primary gatekeeper and two secondary gatekeepers at the organization that participated in my study. All three gatekeepers had access to the participants. Gatekeepers can facilitate contact with participants by using the trust and rapport they have with participants while emphasizing the benefits of the study (Peticca-Harris et al., 2016). I enlisted the help of gatekeepers to identify potential study participants within the organization who met my participant criteria. I emailed information about the study to all participants identified by the gatekeepers prior to involving them in the study.

Interacting with participants requires obtaining informed consent, arranging meeting times and locations, interviewing participants, setting boundaries, avoiding surprises, and maintaining flexibility to accommodate participants (Peticca-Harris et al., 2016). Ethical considerations, trust, disclosure, and impression management are

important factors for gaining access to participants (Greene, 2014). Informed consent of participants is a vital component to upholding the ethical standards and quality of a research study (Sanjari et al., 2014). I sent participants a consent form via email informing them of the study background, procedures, voluntary nature of the study, benefits, risks, and privacy of the study. Participants acknowledged their willingness to participate in the study by replying to my email with the words "I consent." I scheduled 1-hour interviews with participants at a time and place convenient to them.

Maintaining ethical researcher-participant relationships requires acknowledging bias, maintaining rigor, establishing rapport, respecting autonomy, avoiding exploitation, and maintaining confidentiality (Hewitt, 2007). An examination of the personal qualities, values, and beliefs of researchers and the acknowledgment that the context of the researcher and participants may influence the findings may help avoid potential bias from researcher-participant relationships (Hewitt, 2007). Building rapport with participants encourages information sharing and requires researchers to ensure confidentiality, establish trust, and demonstrate respect for participants (McDermid, Peters, Jackson, & Daly, 2014). Autonomy is a key principle of the Belmont Report, which requires treating participants with respect and protecting them with diminished autonomy (Judkins-Cohn, Kielwasser-Withrow, Owen, & Ward, 2014). Autonomy is a critical component of researcher-participant relationships requiring the disclosure of information, voluntary participation, and informed consent of participants (Judkins-Cohn et al., 2014). Avoiding confusion, exploitation, and potential harm to participants requires a clearly defined interview process and unambiguous questions (Hewitt, 2007). Promoting confidentiality

by protecting the privacy of participants improves the researcher-participant relationship and facilitates the sharing of information (McDermid et al., 2014).

I reflected on the interview location and environment to ensure it promoted participant relationships and avoided bias. I welcomed participants and explained the objectives of my study. I reviewed the confidentiality of the study and steps I was taking to maintain the participants' privacy. I ensured participants were given informed consent and ensured participation was voluntary. I invited them to ask questions about the study or my background prior to starting the interview process to avoid any confusion or exploitation.

Research Method and Design

Qualitative case study design allows the researcher to explore participants' perspectives, meanings, and subjective views regarding the research phenomenon (Yilmaz, 2013). This design enables researchers to gain extensive knowledge of the phenomenon and understand the real-world issues associated with the phenomenon (Wynn & Williams, 2012). A qualitative case study allowed me to explore collaboration strategies software development teams use to minimize the amount of technical debt these teams create. A qualitative case study improves a researcher's understanding of the phenomenon and allows him or her to develop a detailed interpretation (Wynn & Williams, 2012). I explored technical debt and collaboration to improve my understanding of these areas allowing me to provide insights for software development leaders.

Method

The three methods of research are qualitative, quantitative, and mixed methods (Venkatesh, Brown, & Bala, 2013). I selected a qualitative method for this study because I was trying to capture participants' perceptions, views, and meanings of collaboration and technical debt in a real-world setting. Qualitative research is an emergent, inductive, and interpretive approach to studying phenomena through the experiences and meanings of others in a natural setting (Yilmaz, 2013). A qualitative researcher explores the phenomenon as viewed and experienced by participants without any preconceived or predetermined beliefs imposed by the researcher (Kemparaj & Chavan, 2013). The perceptions of the participants are one of the most important aspects of a qualitative study (Ritchie, Lewis, Nicholls, & Ormston, 2013). Qualitative researchers study the perceptions, experiences, opinions, beliefs, and values of the participants in their natural setting to generate knowledge and understand their meaning (Ritchie et al., 2013). Qualitative researchers use inductive reasoning to examine the context, interpretation, and meaning of participants' experiences (Yilmaz, 2013).

Qualitative research is often associated with interpretivism, ontology, and epistemology (Aliyu, Bello, Kasim, & Martin, 2014). Researchers with an interpretivist paradigm believe that multiple realities exist and they seek to explore, analyze, and understand these multiple realities (Yilmaz, 2013). The goal of interpretivism is to understand the subjective meanings of participants (Wynn & Williams, 2012). An ontological view allows researchers to understand the life experiences of participants and any shared realities that may exist regarding the phenomenon under study (Aliyu et al.,

2014). A subjectivist epistemology approach permits multiple explanations for a phenomenon by establishing research-participant relationships to understand the phenomenon (Yilmaz, 2013). I selected a qualitative research method to understand the subjective views and participants and the multiple interpretations on collaboration strategies they use to minimize technical debt.

Qualitative researchers often study a small number of participants who can generate a large amount of information and provide a thorough understanding of the phenomenon under study (Yilmaz, 2013). Qualitative research often employs participant observation, in-depth interviews, document analysis, and focus groups as the main methods of data collection (Kemparaj & Chavan, 2013). The results from qualitative research are difficult to generalize due to the highly subjective nature of the study and dependence on the context of the study (Yilmaz, 2013). The researchers are the main instrument of data collection in qualitative research and must distance themselves from the phenomenon under study to prevent any biases from influencing the data (Onwuegbuzie & Byers, 2014). I selected a qualitative research method to study a small population of software development leaders collecting data from in-depth interviews and document analysis.

I explored the possibility of using a quantitative or mixed-methods approach for my research study. I did not choose quantitative methods for my study. Quantitative research focuses on testing theories, determining relationships between variables and measuring numbers to study phenomenon (Yilmaz, 2013). The focus of my study was not testing hypotheses or determining causation so a quantitative study was not appropriate.

Quantitative researchers use statistical analysis to prove or disprove hypotheses to generalize the results to a large population (Barczak, 2015; Hoare & Hoe, 2013). I was not seeking to analyze statistical data or generalize my research so a quantitative study was not suitable. Quantitative researchers collect data through measurements and use the reliability and validity of measures to address validation issues (Venkatesh et al., 2013). I was not collecting data from measurements or validating measures so a quantitative study was not appropriate.

A mixed-methods research approach combines quantitative and qualitative methods in the same research study by including collection, analysis, and interpretation of both numerical and narrative data (Hayes et al., 2013). Mixed-method researchers design, build and test theories in addition to completing inductive and deductive analysis within the same study (Venkatesh et al., 2013). My research focused solely on participants' experiences and not generating theories, testing hypotheses, or studying relationships between variables so a mixed-methods approach was not appropriate. A mixed-methods research approach must describe the integration of findings from multiple methodological components (Fetters, Curry, & Creswell, 2013). A mixed-methods researcher must describe how they managed any inconsistencies or discrepancies between these methods (Fetters et al., 2013). My research did not involve the quantitative principles of hypotheses testing, variable relationships or measurements. I was not integrating multiple research methods so a mixed-methods approach was not appropriate for my study.

Research Design

The major design types for qualitative research are phenomenology, grounded theory, ethnography, case study, and narrative analysis (Yilmaz, 2013). A case study will develop a detailed interpretation of a specific case or multiple cases by studying an event, program or activity (Wynn & Williams, 2012). I chose a case study design to acquire a thorough understanding of how software development leaders use collaboration activities to minimize technical debt reducing costs within their organization. A case study design improves a researcher's understanding of a phenomenon by performing a comprehensive examination to investigate and explore a multifaceted phenomenon in a real-world setting asking how or what type questions (Cronin, 2014; Wynn & Williams, 2012). I chose a case study design to conduct a thorough inquiry into the complex activity of using collaboration strategies to minimize technical debt.

Case studies may focus on individuals, groups, activities, relationships, interactions, specific phenomenon or anything else the researcher considers necessary (Wynn & Williams, 2012). I chose a case study design to focus on software development leaders and the relationship between collaboration strategies and technical debt. The fundamental predisposition of a case study is to understand the context of one or more decisions, the implementation of those decisions and the results of those decisions (Cronin, 2014). I chose a case study design to explore decisions made by software development leaders to use specific collaboration strategies, the implementation of these strategies and the influence of these strategies on technical debt.

A case study is a detailed inquiry into a specific and complex phenomenon within a real-world context (Yin, 2013). The key tenet of a case study is to explore an event or phenomenon in depth and in its natural setting (Wynn & Williams, 2012). A case study researcher attempts to understand the distinctiveness of an individual case in a setting (Abma & Stake, 2014). I chose a case study design to explore a specific case of software development leaders using collaboration strategies to minimize technical debt in a real-world organization within the natural setting of a software development team.

A phenomenological research design studies the human experience from the perspective of the participants living through the phenomenon (Hanson et al., 2011). Roberts (2013) notes the purpose of a phenomenological study is to describe the lived experiences of participants. A phenomenological researcher needs to understand how the social, cultural, and political contexts in their everyday lives influence participants' realities (Tuohy, Cooney, Dowling, Murphy, & Sixsmith, 2013). Interpretation by participants of their experiences are critical to answering my research question, but the social, cultural, and political aspects of their world are not. While some aspect of a phenomenological design was appropriate for this study, I did not consider it the most appropriate design for my study.

Ethnographical research focuses on the links between cultures and peoples (Astalin, 2013). An ethnographical research design seeks to describe cultural behavior through participant observation where the researcher is actively involved and is essential for understanding the culture under study (Cruz & Higginbottom, 2013). An ethnographical researcher explores participants' behaviors rather than their perceptions or

views (Walker, 2012). I did not choose an ethnographical study because it focuses on observing cultures and behaviors within a group of people. I did not believe these are major factors influencing collaboration's role in technical debt. An ethnographical study also relies heavily on observations that I believe would not have provided enough information for a thorough analysis of the data.

A narrative researcher explores the life experiences of participants who recount their experience with the phenomenon to the researcher (Wolgemuth, 2014). Caine, Estefan, and Clandinin (2013) noted that a narrative design collects data about how participants view themselves and their experience with an event. Participants tell their story as they remember it. A narrative research design would not be appropriate because it explores the life of an individual (Walker, 2012) and this study focused on collaboration between multiple individuals. Studying individuals would not have yielded the appropriate data to answer my research questions. My research focused on collaboration between participants requiring my understanding their perceptions and not their stories.

The concept of data saturation involves adding new participants continually into a study until a researcher finds no new data and there is sufficient data to replicate the research study (Marshall, Cardon, Poddar, & Fontenot, 2013). A study reaches data saturation when additional data collection no longer contributes new data to the study (Lincoln & Guba, 1985). I collected data until no new information is being generated indicating I had achieved data saturation. I achieved data saturation by using census sampling to collect data from everyone in the study population.

Interviewing participants with significant knowledge and experience with the phenomenon under study assists a study achieving data saturation (Malterud, Siersma, & Guassora, 2015). I interviewed participants who have significant knowledge and experience using collaboration strategies to minimize technical debt in software development to assist in reaching data saturation. Face-to-face interviews facilitate a study reaching data saturation by asking multiple participants the same probing questions to ensure richness and depth of the data collected (Fusch & Ness, 2015). I asked all participants the same probing questions ensuring the depth and richness of data I collected to help reach data saturation.

Data triangulation of multiple data sources and the depth of data collected from multiple data sources is a means to achieve data saturation (Fusch & Ness, 2015). I collected responses to my interview questions from everyone in my study population. I collected observations from participants during the interviews. I collected data from documents and other artifacts relating to software development standards, methodologies, best practices, collaboration and processes from the organization. I tracked the data I collected from these multiple sources to facilitate triangulation to determine when I achieved data saturation. I collected data until no new information was being generated indicating I had achieved data saturation.

Population and Sampling

The population of my study consisted of senior software development leaders from two software development teams within a single line of business at a large healthcare provider in the state of California. The population characteristics in a

qualitative study relate to participants' subjective experience with the phenomenon of interest in the study (Stern et al., 2014). The population of my study all had experience using collaboration strategies in software development to minimize technical debt, which was the phenomenon of my study. The first step in the data collection process is to define the study population by using inclusion and exclusion criteria (Robinson, 2014). The study population included software developers with 10 years of experience, who held a senior leadership position within the organization and actively participate in one or more collaboration activities. The more criteria used to define the population, the more specific these criteria are and the number of domains increases the homogeneity of the study population (Robinson, 2014).

The site and setting of an interview are important factors that might influence the content of an interview and affect data collection (Doody & Noonan, 2013; Vahasantanen & Saarinen, 2013). The interview setting should be in a place that is free from interruptions to avoid distracting participants, which might affect data collection (Doody & Noonan, 2013). I conducted interviews in a conference room located at the organization that I reserved for my exclusive use. I closed the conference room door and hung a sign requesting not to be disturbed to avoid distractions from individuals not participating in the interviews. I closed all blinds on the windows to avoid any outside distractions. I ensured the room does not have any unusual smells. I covered pictures, signs, and other decorations that might be distracting. The interview setting should minimize background noises that may distract participants or interfere with audio recordings that may affect data collection (Dikko, 2016). I ensured there were no

background noises within the conference room or anywhere nearby that might have distracted my participants or interfered with my interviews. The place and time of the interview must be convenient to participants and allow adequate time to conduct the interview (Dikko, 2016). I used a conference room at the workplace of my participants to minimize any loss of time due to traveling to and from the interview location.

The three ways to select participants from the study population are census, probability sampling and nonprobability sampling (Lucas, 2014). I used a census sampling strategy to interview all individuals in my study population. A census involves selecting everyone in the study population. (Kish & Verma, 1986; Lucas, 2014). The primary objective of a census is to collect complete and detailed data regarding the study phenomenon from the population (Kish & Verma, 1986). A census is more suitable for smaller finite populations where time and cost are less of a concern (Jordan, 2013). The population for my study was small and finite so a census sampling strategy was the best option to provide a complete, detailed understanding of the phenomenon.

My study population consisted of 13 senior software development leaders from two software development teams from a single line of business within the organization who met all the eligibility criteria. I invited the entire study population to participate in my study. An important aspect of sampling is to ensure the knowledge gained is representative of the study population to draw reliable conclusions (Etikan, Musa, & Alkassim, 2016). A researcher must scrutinize the characteristics of a sample to determine how well the sample represents the study population (Sedgwick, 2013). My sample was the entire study population so it completely represented the study population

allowing me to draw reliable conclusions. A census sample involves the entire population so it perfectly generalizes to that population (Kish & Verma, 1986; Lucas, 2014). My sample was the entire study population allowing me to generalize my findings to the population.

A census is subject to response errors and the key measure of quality in a census is the level of response achieved. (Bell et al., 2015). A census may be prone to duplication or omission of participants (Kish & Verma, 1986). I kept a log of all participant interviews to ensure I interviewed everyone in the study population. I used the log to track follow-up interviews during the member checking process to ensure that I member checked interview data with all participants. Everyone participates with a census strategy so there is a little loss of confidentiality when compared with other sampling strategies where participants are a subset of the population (Kish, 1979). I protected the confidentiality and privacy of the entire study population by replacing any identifying information in interview data with codes to conceal participants' identity.

The concept data saturation involves adding new participants continually into the study until no new data appears and there is sufficient data to replicate the research study (Fusch & Ness, 2015; Marshall et al., 2013). Malterud et al. (2015) contend the sample size necessary for data saturation is dependent on how well the sample represents the population, the structure of the interviews, the quality of the interviews and the participants' knowledge and experience with the phenomenon under study. Interviews facilitate a study reaching data saturation by asking multiple participants the same

probing questions to ensure richness and depth of the data collected (Fusch & Ness, 2015).

Data saturation is a standard and elusive component of qualitative research with few concrete guidelines and different meanings to various researchers (Marshall et al., 2013). Data triangulation of multiple data sources is a means to achieve data saturation, enhance the reliability of this study and ensure the validity of this study (Fusch & Ness, 2015). I collected data from documents and other artifacts relating to software development standards, methodologies, best practices, collaboration and processes in the organization in addition to the observations and responses from my interviews. I tracked the data I collected from these multiple sources to facilitate triangulation to determine when I achieved data saturation. I collected data until redundancy of the data had occurred with no new information being generated indicating I had achieved data saturation.

Ethical Research

Ethical issues may occur in any part of a research study at any time (Qu & Dumay, 2011). Ethical challenges may arise from researcher-participant relationships, research design, data collection, data analysis, interpretations or anywhere else in a research study. Qualitative researchers have a responsibility to uphold ethical standards and principles while conducting a study (Haahr, Norlyk, & Hall, 2014). I ensured all my processes and procedures were ethical. Sanjari et al. (2014) noted there are potential ethical challenges between researchers and participants in qualitative studies and suggested researchers follow a protocol while conducting their studies. Qu and Dumay

(2011) contend that following an interview protocol helps protect participants in a study. I followed a protocol placing great emphasis on protecting participants in this study by ensuring they came to no harm, were willing volunteers, could withdraw at any time, understood the intent of the research and I protected their privacy and confidentiality.

Institutional review boards (IRBs) evaluate the risks, benefits, subject selection methods, informed consent process, and methods for protecting privacy and confidentiality in research studies (Cook, Hoas, & Joyner, 2013). I obtained approval from the Walden University Institutional Review Board (IRB) prior to collecting any data or conducting any interviews. IRBs require consent forms in all research involving human subjects and the text in consent forms should match the reading level of participants (Ferreria, Buttell, & Ferreria, 2015). All participants in my study acknowledged their willingness to participate in my study by confirming their consent in accordance with the IRB guidelines. The consent form provided information on the intent of the study, benefits, risks, confidentiality, and right to withdraw. Researchers should notify their IRB in the event the study's research design changes or the researcher encounters unexpected results (Hanson et al., 2011). I completed the online course on protecting human research participants with the National Institutes of Health and received a certificate of completion (see Appendix A).

Informed consent is an ethical requirement of qualitative research outlining the researcher's responsibility to inform participants of varying aspects of the study (Sanjari et al., 2014). The participants for this study were voluntary and able to withdraw from the study at any time. I immediately destroyed any data collected from participants who

withdraw from the study. I used census sampling to interview everyone in my study population so replacing participants who withdrew was not applicable since I was already interviewing everyone. All participants in a qualitative case study should sign an informed consent form confirming their willingness to participate (Qu & Dumay, 2011). An informed consent form notifies participants about the nature of the study, procedures, participation is voluntary, risks, benefits, and confidentiality of the study (Judkins-Cohn et al., 2014). I required all participants to acknowledge their willingness to participate in the study by replying to my email with the words "I consent" prior to their participation in the study. Participants receiving compensation for their participation may lead to inaccurate data (Robinson, 2014). I did not compensate participants to solicit their participation.

The confidentiality of participants is required to uphold the ethical standards of a study and may be beneficial to a study (Beskow, Check, & Ammarell, 2014). A researcher maintains confidentiality in a study by concealing the identity of participants and using codes instead of names to link participants to their interview data (Ferreria et al., 2015). The intentional or unintentional disclosure of confidential information may cause harm to participants (Ferreria et al., 2015; Mealer & Jones, 2014). A researcher should keep confidential data in either paper or electronic form in a secure storage device and destroy the data after the researcher no longer needs the data (Mealer & Jones, 2014). All electronic data collected is stored on an encrypted, secure storage media. All hard copy documents are stored in a securely locked safe. I will retain all data for at least five years to protect the participants' confidentiality. After five years, I will permanently

delete all electronic media and shred any hard copy documents. I coded all data to protect the identities of all participants and organizations so they remain anonymous. I conducted interviews in a private and confidential manner. I have not released any identifying information such as names, e-mails, address or phone numbers. All potential participants received an invitation to participate in the study and an informed consent form detailing these privacy and confidentiality steps.

Data Collection

Instruments

The main methods of data collection in qualitative research include participant observation, interviews, and document analysis (Yilmaz, 2013). The researcher is the primary instrument of data collection in qualitative research studies (Lincoln & Guba, 1985). A researcher should be an informant-centered instrument in qualitative research viewing participants as the experts allowing them to share the information they deem most important (Peredaryenko & Krauss, 2013). The researchers' role includes investigating varying points of view and perspectives during the data collection process (Kavoura & Bitsani, 2014).

I developed informative interview questions, listened to participant responses without interruptions, was flexible during the interview process, and utilized follow-up questions to ensure I fully understood all participants' points of views and perspectives. I sought the guidance of participants when selecting documents to collect and analyze. I viewed participants as the authorities on my research phenomenon and encouraged information sharing whenever possible.

A human instrument in qualitative research is susceptible to subjectivity, predispositions, and biases (Peredaryenko & Krauss, 2013). A researcher may inject bias during data collection due to flaws in research instruments, improper sample selection, collecting insufficient data, influencing participants, subjective interpretation, and analysis of data or otherwise allowing personal beliefs to influence the research study (Onwuegbuzie & Byers, 2014; Roulston & Shelton, 2015). Strategies for managing bias involve researchers being aware of their own values, predispositions, biases, and subjectivity in the research process (Roulston & Shelton, 2015). Maintaining a journal of self-reflection and self-examination throughout a study reflecting on the degree to which their biases and subjectivity might have influenced the various components of the study may help avoid bias (Onwuegbuzie & Byers, 2014).

I maintained a reflective journal throughout the research process consciously writing about any personal values, beliefs, biases or subjectivity that might have influenced my study. I examined, reflected, and interviewed myself throughout the data collection process looking for inadequacies, undue influence, subjective interpretations, and other forms of bias. I wrote about how I counteracted my biases and how my counteractions might have led to bias.

Interviews. Research interviews are one of the most important methods for collecting data in qualitative research (Qu & Dumay, 2011). The purpose of semistructured interviews was to ascertain subjective responses from participants regarding a specific phenomenon and usually follows a detailed interview protocol (McIntosh & Morse, 2015). Asking open-ended questions in an organized manner will

help generate detailed descriptions and probe for deeper insight to improve the validity of a study (Hanson et al., 2011). The semistructured interview involves predetermined open-ended questions, allows the researcher to seek clarification and allows the researcher to explore new topics that emerge (Doody & Noonan, 2013). A researcher organizes semistructured interviews around the main topic but is flexible to allow participants to change direction or inject new themes (O'Keeffe, Buytaert, Mijic, Brozovic, & Sinha, 2016).

I conducted semistructured interviews consisting of a set of predetermined, open-ended questions (see Appendix B). I asked follow-up questions as necessary during the interviews to get clarification on participants' responses. I maintained flexibility in the interviews allowing participants to change direction and inject new topics.

The research questions serve as a basis for designing interview questions and researchers should develop interview questions based on what information will help answer the research questions (Anfara, Brown, & Mangione, 2002). Researchers must develop as much expertise as possible in relation to the research topic areas so they can develop informed questions (Qu & Dumay, 2011). Correlating interview questions with research questions by cross-referencing each interview question to the study's research questions is crucial for validating interview questions (Anfara et al., 2002). Using a matrix to map interview questions to research questions will help align interview questions to research questions to identify any gaps in the interview questions (Castillo-Montoya, 2016).

I conducted a literature review to increase my knowledge and expertise in the areas of collaboration, technical debt, and the TAM. My research question and conceptual framework formed a basis for developing my interview questions. I created a matrix (see Appendix D) to map interview questions to my research topic areas ensuring there were no gaps and my interview questions align to my research questions.

An interview protocol is a set of rules and guidelines a researcher uses while conducting interviews in a qualitative study (Dikko, 2016; Jacob & Furgerson, 2012). An interview protocol is an instrument aligned to a study's purpose facilitating inquiry-based conversations (Castillo-Montoya, 2016). An interview protocol may contain preinterview scripts, postinterview scripts, interview questions and interview reminders (Jacob & Furgerson, 2012). Researchers can mitigate potential ethical challenges between researchers and participants in qualitative studies by following research protocols (Sanjari et al., 2014). An interview protocol helps protect participants in a study by ensuring participants' personal details are kept private (Qu & Dumay, 2011).

I used an interview protocol (see Appendix B) that will consist of preinterview activities, interview reminders, interview questions, and postinterview activities. My preinterview activities consisted of an introduction, verification of informed consent, reminding participants about recording audio, and confidentiality. The main portion of interview started with turning on the audio recording device, stating the participant's identifying code, stating the date and time, asking my interview questions, asking the participant to share any other relevant information, and stopping the audio recording. My

postinterview protocol explained the concept of member checking, scheduling a follow-up interview, thanking participants, and providing my contact information to participants.

Member checking. Member checking is the most important technique for assessing the validity and enhancing rigor in qualitative research (Lincoln & Guba, 1985). Member checking is an important method for establishing the dependability and reliability of a study by allowing participants to confirm the researchers' interpretation of the data collected from participants (Onwuegbuzie & Byers, 2014). Member checking is the process where researchers ask participants to confirm the researchers' interpretations of a participant's experiences, meanings, and viewpoints (Koelsch, 2013). Member checking is a continual process of analyzing and interpreting emerging themes from the data, presenting the interpretations to participants, asking participants to confirm the interpretations, and asking participants follow-up questions for additional clarification (Birt, Scott, Cavers, Campbell, & Walter, 2016). A researcher continues member checking until the participants confirm all interpretations, the participants provide no new information, and additional clarification is no longer required (Harvey, 2015).

I used member checking to increase the validity, reliability, and dependability of my study by confirming my interpretations of emerging themes from the data I collected from participants. After my interview with participants, I scheduled follow-up interviews with participants for member checking. I emailed participants my interpretations of their data a few days prior to the follow-up interview. During the follow-up interview, I asked participants to confirm whether my descriptions, interpretations, and understandings from their interview accurately reflects their experiences, meanings, and viewpoints. I asked

participants follow-up questions as necessary to seek clarification of the data. I continued scheduling follow-up interviews and member checking until participants confirmed all my interpretations.

Document analysis. Document analysis involves the selection of written or recorded materials that confirm events or provide explanations and descriptions (Lincoln & Guba, 1985). Document analysis is an essential instrument of qualitative research that provides the researcher a better understanding of the research phenomenon (Islam, 2014). Case study researchers use information found in documents to corroborate information from and add context to other data sources (Boblin, Ireland, Kirkpatrick, & Robertson, 2013). Additional sources of data allow researchers to check for consistency and triangulation of data to improve the reliability of research (Patton, 1999). Key informants can assist researchers in locating documents important to the research study (Boblin et al., 2013). Document analysis can suffer from low retrievability, selection bias, and access restrictions (De Massis & Kotlar, 2014). Researchers must ensure the quality and trustworthiness of artifacts and documents they use in research (Islam, 2014).

I used document analysis to collect data on collaboration and technical debt. I analyzed procedural documentation related to software development standards, methodologies, best practices, collaboration, and processes. I was the primary instrument for collecting and reviewing all documentation. The goal of my document collection was to measure the perceptions of my participants to the organization's strategies regarding collaboration and technical debt management. The documentation provided an additional source of data to corroborate and check the consistency of other data sources.

Researchers must select a wide range of sources for documents to ensure a comprehensive understanding of the data (Dunne, Pettigrew, & Robinson, 2016). The process of selecting and analyzing documents requires a researcher to assess the quality and trustworthiness of the documents (Dunne et al., 2016). The quality and trustworthiness of a document depend on the authenticity, credibility, reliability, and accuracy of the document (Donaldson, 2016). Determining the authenticity of a document requires the researcher to understand the source of the document and judge the content on this basis (Dunne et al., 2016). A researcher determines the credibility of a document by questioning the perspective of the author and the reason for creating the document (Dunne et al., 2016). The accuracy to which a document reflects the contents or events determines the reliability of a document (Donaldson, 2016).

I selected a wide range of documents with the help of key informants from varying sources to obtain a more thorough understanding of the data. I loaded these documents into the Qiqqa research management software so that I may identify themes and add annotations to each document. I used the Qiqqa software to explore and brainstorm the documents looking for common themes. I used the Qiqqa document insight features to identify common keywords and themes within the documents. I used the Qiqqa brainstorming features to create mind maps, identify links between documents and arrange documents to help identify important themes in the documents. I judged the quality and trustworthiness of the documents by determining the authenticity, credibility, and accuracy of the documents. I investigated the source of all documents, determine the reason for the creation of the documents and determine if the documents accurately

reflect the contents. I determined if the actual contents of the documents match the authors' purpose and the reason for creating the documents.

Triangulation. Denzin (1978) argued that a single method of data analysis is not adequate to describe a research phenomenon and the use of multiple methods to collect data improves the validity and reliability of a study. Methodological triangulation entails the use of multiple methods of data collection and analysis regarding the same phenomenon to develop a thorough understanding of the phenomenon contributing to greater accuracy, reliability, and validity (Denzin, 1978; Hussein, 2015). Methodological triangulation benefits studies by providing richer data, correlating the data, increasing validity and enhancing understandings of the research phenomena (Fusch & Ness, 2015). Within-method triangulation is a type of methodological triangulation involving crosschecking complementary data collection methods within a qualitative study increasing the consistency and credibility of a study (Denzin, 1978; Hussein, 2015). Methodological triangulation involves comparing data collected from varying methods to enrich the data, confirm or refute findings and further explain findings (Patton, 1999).

I used the within-method type of methodological triangulation to analyze the qualitative data I collected through semistructured interviews, interviews observations and document analysis. I crosschecked data collected from semistructured interviews with my interview observation data and document analysis. I looked for consistencies within the data that may provide confirmations of my interpretations. I looked for data in one method that may enrich the data or further explain the data collected from another

method. I looked for inconsistencies in the data and attempt to develop reasonable explanations for the differences.

Data Collection Technique

Interviews and document analysis are the main methods of data collection in my study. My data collection technique involved gaining access to participants, obtaining informed consent from participants, selecting an interview location, scheduling interviews, following an interview protocol, and conducting member-checking activities. I ensured I had IRB approval before beginning any recruitment or data collection activities.

An important step in gaining access is to obtain approval from key stakeholders during the study design and planning stage (Peticca-Harris et al., 2016). Gatekeepers are individuals who can facilitate gaining access to participants by endorsing a researcher's work (Crowhurst, 2013). Gatekeepers can facilitate contact with participants by using the trust and rapport they have with participants while emphasizing the benefits of the study (Peticca-Harris et al., 2016). I enlisted the help of three upper management gatekeepers to help identify potential study participants within the organization that meet my participant criteria. I emailed the gatekeepers my criteria for participating in my study to help them identify individuals. I had the gatekeepers send me the contact information of all individuals that met my participation criteria who might be willing to participate in my study. I emailed detailed information about my study to all potential participants identified by the gatekeepers. I ensured those willing to participate met all my eligibility criteria.

Informed consent is an ethical requirement of qualitative research outlining the researcher's responsibility to inform participants of varying aspects of the study (Sanjari et al., 2014). An informed consent form notifies participants about the nature of the study, procedures, participation is voluntary, risks, benefits, and confidentiality of the study (Judkins-Cohn et al., 2014). All participants in a qualitative case study should sign an informed consent form confirming their willingness to participate (Qu & Dumay, 2011). I emailed participants a consent form informing them of the study background, procedures, voluntary nature of the study, benefits, risks, and privacy of the study. I require all participants to acknowledge their willingness to participate in the study by replying to my email with the words "I consent" prior to their participation in the study. The participants for this study will be voluntary and able to withdraw from the study at any time. I immediately destroy any data collected from participants who withdrew from the study. I used census sampling to interview everyone in my study population so replacing participants who withdraw is not applicable since I was interviewing everyone.

Researchers should let participants know the time, place, and duration of interviews when scheduling interviews so participants allocate enough time to avoid disruptions (Peticca-Harris et al., 2016). The place and time of the interview must be convenient to participants and allow adequate time to conduct the interview (Dikko, 2016). When conducting multiple interviews on the same day, a researcher should allow 30 to 60 minutes between interviews to decompress and debrief (Rimando et al., 2015). I scheduled face-to-face interviews with each participant lasting about one hour at a place convenient to them. I emailed the participants the agreed upon time, place, and duration

of the interview. I conducted three interviews per day on average making sure there was at least 30 to 60 minutes between interviews.

The site and setting of an interview are important factors that might influence the content of an interview and affect data collection (Doody & Noonan, 2013; Vahasantanen & Saarinen, 2013). The interview setting should be in a place that is free from interruptions to avoid distracting participants, which might affect data collection (Doody & Noonan, 2013). The interview setting should minimize background noises that may distract participants or interfere with audio recordings that may affect data collection (Dikko, 2016). I conducted interviews in a conference room at the organization that I reserved for my exclusive use to ensure privacy and minimize travel time for participants. I closed the conference room door during my interviews and put a notice on the door requesting no disturbances to avoid distractions from individuals not participating in the interviews. I closed all blinds on the windows to avoid any outside distractions. I ensured the room does not have any unusual odors or smells. I covered any pictures, signs and other decorations that might be distracting. I ensured there were no background noises within the conference room or anywhere nearby that might have distracted my participants or interfered with my interviews.

Taking notes during an interview can be distracting, interfere with the interview and detract from a researcher's ability to actively listen to participants and ask probing questions (Onwuegbuzie & Byers, 2014). An audio recording of an interview allows a researcher to relisten to all or parts of an interview to increase their familiarity with the interview allowing better interpretation of the data (Gale, Heath, Cameron, Rashid, &

Redwood, 2013). A researcher should use good quality audio recording equipment and be familiar with its operation (Doody & Noonan, 2013). Using multiple devices when recording audio from interviews provides redundancy in case one device fails (Onwuegbuzie & Byers, 2014). Transcribing audio allows the research to immerse themselves in the data and become familiar with the entire interview (Gale et al., 2013). A disadvantage of audio recordings is the inability to capture impressions, environmental contexts, behaviors, and other nonverbal cues (Sutton & Austin, 2015). I used two, fully charged audio devices to record the interviews for accuracy with the permission of the participants. I took notes during the interviews of my observations of participants' body posture, facial expressions, and hand gestures that may justify further exploration of areas using probing questions. After the interviews, I listened to the audio multiple times to increase my familiarity with the interviews. I transcribed the audio recordings into a Microsoft Word document using the speech recognition software in Windows 10. I removed any personally identifiable information from the transcription and used codes to identify participants.

Reflection helps researchers examine and evaluate research methods, frameworks, and assumptions providing more insight into the study facilitating a deeper level of questioning (Peredaryenko & Krauss, 2013). A reflective journal represents a researcher's first-hand experience with the data and helps researchers understand first perceptions of data (Lamb, 2013b). Recording the researchers' thoughts before and after interviews allows the researchers to identify and reflect on any potential bias that may have influenced a participant's response and skewed the data (Hoare & Hoe, 2013). A

reflective journal allows a researcher to expand ideas, develop new understandings, and draw conclusions from the research by reflecting on the research process and data (Peredaryenko & Krauss, 2013). A reflective journal is an additional source of data providing additional evidence on emerging themes (Lamb, 2013a). Some researchers argue that journal writing lacks objectivity and may overemphasize the researcher's values and beliefs instead of participants (Lamb, 2013b).

I used a journaling approach to record my methods, experiences, observations, thoughts, and judgments throughout the research study. I used a journaling approach to record my thoughts before and after each interview to prevent bias in my data. I used a Microsoft Word document to maintain my reflective journal. I logged my initial thoughts normally in the document. I used the review tracking and commenting features to reflect on my thoughts. This enabled me to keep my initial thoughts and my thoughts as they evolved. I periodically reflected on my journal notes and initial perceptions to expand my understanding of the research. I questioned myself in regards to the methods I used and assumptions I made during the research process.

An interview protocol is a set of rules and guidelines a researcher uses while conducting interviews in a qualitative study (Dikko, 2016). An interview protocol may contain preinterview scripts, postinterview scripts, interview questions, and interview reminders (Jacob & Furgerson, 2012). An interview protocol is an instrument aligned to a study's purpose facilitating inquiry-based conversations (Castillo-Montoya, 2016). I used an interview protocol (see Appendix B) to facilitate my interview processes. I introduced myself to the participant and thanked them for participating. I verified receipt of the

consent form, answered any questions or concerns they had. I reminded participants that I would be recording the interview and that the interview would remain strictly confidential. I turned on the recording device and announced the date, time, and identifying code of the participant. I asked each question in my interview protocol (see Appendix B) starting with the first question and continuing through to the last question. I allowed the participant to respond to each question and asked additional probing questions as necessary. After I had asked all my questions, I asked the participant if they wanted to share any more information about the topics. I asked the participant if they were aware of any documentation that might be relevant to the topics discussed. I explained the concept of member checking and scheduled a follow-up interview to review my interpretations with them. I discontinued the audio record by turning off the device and thanked the participant for partaking in the study. I confirmed the participant had my contact information for any follow-up questions and concerns.

Member checking is the process where researchers ask participants to confirm the researchers' interpretations of a participant's experiences, meanings, and viewpoints (Koelsch, 2013). Member checking is a continual process of analyzing and interpreting emerging themes from the data, presenting the interpretations to participants, asking participants to confirm the interpretations and asking participants follow-up questions for additional clarification (Birt et al., 2016). A researcher continues member checking until the participants confirm all interpretations, the participants provide no new information, and additional clarification is no longer required (Harvey, 2015).

After the conclusion of my interview with the participant, I scheduled a follow-up interview with the participant for member checking. Prior to the follow-up interviews, I analyzed the data collected from the participant to develop my interpretations of the data they provided and any emerging themes. I developed rich descriptions of the data encompassing my interpretations and understandings of their interview data. I emailed the participant my interpretations of their data a few days prior to the follow-up interview. During the follow-up interview, I asked the participant to confirm whether my descriptions, interpretations, and understandings of their interview accurately reflects their experiences, meanings, and viewpoints. I also asked the participant follow-up questions as necessary to seek clarification of my understanding of their views. I asked the participant if they had any new data to share with me. If I receive new information from the participant, I scheduled an additional follow-up interview and repeated the member checking process. I kept repeating this member checking process if I kept receiving new information from participants.

Case study researchers use information found in documents to corroborate information from and add context to other data sources (Boblin et al., 2013). Researchers must select a wide range of sources for documents to ensure a comprehensive understanding of the data (Dunne et al., 2016). Key informants can assist researchers in locating documents important to the research study (Boblin et al., 2013). Document analysis can suffer from low retrievability, selection bias, and access restrictions (De Massis & Kotlar, 2014).

I selected a wide range of documents with the help of key informants from varying sources to obtain a more thorough understanding of the data. I enlisted the help of three upper management gatekeepers to help identify and gain access to documents related to my study. I asked key informants for any documentation related to software development standards, methodologies, best practices, collaboration, technical debt or other software development processes. After each interview, I asked the participants to identify any documentation related to the interview questions. I requested assistance from my three gatekeepers to facilitate access to documents identified by participants. I emailed the key informants who identified the documents to determine the source and purpose of all documents. I asked these key informants if the actual contents of the documents match the authors' purpose and the reason for creating the documents.

Data Organization Techniques

Organizing data in qualitative research is an essential process to understanding and analyzing the data collected adding to the trustworthiness of the study (Elo et al., 2014). A researcher chooses codes, concepts, and categories to facilitate the labeling, sorting, and comparison of the data collected (Vaismoradi, Jones, Turunen, & Snelgrove, 2016). Categorization strategies often involve analyzing similarities in data and grouping data by likeness (Plamondon, Bottorff, & Cole, 2015). I assigned each participant a code ranging from one to eight to maintain their confidentiality and track their data. I used high-level folders in my encrypted cloud storage to organize participant information, interview data, audio recordings, member checking data, organization artifacts and my

research journals. I used multiple folders within the high-level folders to enhance the categorization of the data and artifacts.

A reflective journal allows a researcher to expand ideas, develop new understandings and draw conclusions from the research by reflecting on the research process and data (Lamb, 2013b). Vicary, Young and Hicks (2016) argue that reflexivity, reflection and journaling improve the quality and validity of qualitative data. A research journal is an important aspect of documenting the research improving the validity of the study and providing a means for other researchers to judge the transferability of the research (Lamb, 2013a). I used a journaling approach to record my methods, experiences, observations, thoughts, and judgments throughout the research study. I periodically reflected on my journal notes and initial perceptions to expand my understanding of the research. I questioned myself in regards to the methods I used and assumptions I made during the research process.

Qualitative researchers often use a categorization matrix to identify, track, and group data corresponding to the concepts and categories relating to the data to improve the validity of the study (Elo et al., 2014). I used an Excel spreadsheet to associate information, forms, journal notes, and research data with participant codes. I used the spreadsheet to assign concepts and categories to the data allowing me to sort and group the data to improve my understanding of the data.

All electronic data collected will be stored on an encrypted, secure storage media and stored along with any hard copy documents in a securely locked safe to protect the participants' confidentiality. I will retain all data for a period of at least five years from

the publication date of this study. After five years, I will permanently delete all electronic media and shred any hard copy documents.

Data Analysis Technique

The purpose of my data analysis was to repeatedly search the data I collected until I had a meaningful answer to my research question on collaboration strategies software development leaders use to minimize technical debt. Data analysis is an iterative process to systematically search and assemble data in a meaningful manner allowing ideas to develop (Noble & Smith, 2014). The data analysis process reduces the amount of data collected by grouping the data into categories and seeking to understand the meaning of the data (Bengtsson, 2016). The data analysis must be a transparent, meticulous, and methodical process constructing an accurate description of phenomena from participants' views (Noble & Smith, 2014). I sought to understand the views expressed by the participants and interpret their meaning in a reliable manner. Qualitative data analysis is an inductive process focusing on the meaning of the data and allowing concepts to emerge from the data (Noble & Smith, 2014). An inductive approach to data analysis consists of coding data by creating categories, interpreting the data, and checking the trustworthiness and representativeness of the data (Elo et al., 2014).

I used the within-method type of methodological triangulation to analyze the qualitative data I collected through semistructured interviews and artifacts pertaining to software development standards, methodologies, best practices, collaboration, and processes. Methodological triangulation entails the use of multiple methods of data collection and analysis regarding the same phenomenon to develop a thorough

understanding of the phenomenon contributing to greater accuracy, reliability, and validity (Denzin, 1978; Hussein, 2015). Methodological triangulation benefits studies by providing richer data, correlating the data, increasing validity, and enhancing understandings of the research phenomena (Fusch & Ness, 2015). Denzin (1978) argued that a single method of data analysis is not adequate to describe a research phenomenon. The purpose of this type of within-method triangulation is to use complementary data collection and analysis methods to increase the accuracy and credibility of a study (Hussein, 2015).

The first step in my coding process was immersing myself in the data by repeated readings of the data so I was familiar with the depth and breadth of the data. I took notes throughout the coding process to track my ideas and reflect on my analysis of the data. I used multiple coding methods to generate a list of initial codes that best represent the data and are consistent with my research question. I documented in my journal the importance of each code I created and reflected upon any ideas I had regarding the code. I performed multiple cycles of coding using multiple coding methods to look for explanations, patterns, relationships, and underlying meanings of the data. I categorized and grouped codes during this process to develop themes and organize them into various domains of knowledge. I repeated this process as necessary to ensure I had meaningful explanations of the phenomena consistent with my research question.

Computer-assisted qualitative data analysis software (CAQDAS) assists researchers with multiple types of data analysis allowing underlying relationships to emerge and provide better results than manual analysis (Moylan, Derr, & Lindhorst,

2015). The NVivo 10 software can perform constant comparison analysis, keywords in context, word count, classical content analysis, domain analysis, taxonomic analysis and componential analysis (Castleberry, 2014). I used the NVivo 10 and Qiqqa software to analyze the various sources of data I collected from interviews, notes, audio transcripts and documents by using the multiple coding methods available in the software to code, categorize and group my data. I used the memo and notes functionality in the software as a research journal to track my ideas, assumptions, and reflections.

Qualitative data requires a systematic coding process to catalog data into themes allowing researchers to interpret subjective data in a valid and reliable manner (Bernauer, Lichtman, Jacobs, & Robinson, 2013). Researchers can use the advanced tools in NVivo to visualize data through the use models, graphs, reports, maps, and cluster analyses to monitor emerging themes (Edwards-Jones, 2014). I used NVivo and Qiqqa to generate word trees, word clouds, mind maps, cluster analyses, cluster maps, and graphs to provide a visual representation of my data allowing me to interpret context, relationships, frequencies and find emerging themes. I sorted, arranged, assembled, and analyzed the data repeatedly until major themes and trends emerge that are consistent with my research question. I searched for familiar patterns and recurring themes that might indicate a correlation between software development, collaboration, strategies, technical debt, and characteristics of the TAM.

My data analysis included data from my literature review that I found relevant to my research question, conceptual framework or data I collected from interviews and documents. I also searched for newly published studies that may be relevant to my

research question, conceptual framework or data I collected. I included any new studies I found in my data analysis.

Reliability and Validity

A qualitative research study should be trustworthy, credible, dependable, original, and robust (Yilmaz, 2013). The criteria for assessing the trustworthiness of qualitative research are credibility, transferability, dependability, and confirmability, which are equivalent to the quantitative principles of internal validity, external validity, reliability, and objectivity (Lincoln & Guba, 1985).

A qualitative study is reliable if other researchers could produce similar results using the same methods, techniques, and phenomena (Zohrabi, 2013). Reliability of research data is a prerequisite for the validity of the research data (Stevens, Lyles, & Berke, 2014). It can be difficult to replicate a qualitative study due to the subjective nature of the researcher and participants. Zohrabi (2013) contends researchers should focus on dependability and consistency of the data rather than reproducibility of the results. Researchers should ensure the findings and results are consistent and dependable based on the data collection processes. The prevailing strategy is to ensure reliability by providing transparent and detailed descriptions of all methods, procedures, techniques, and phenomena so other researchers could hypothetically produce the same results (Stevens et al., 2014).

The validity of a qualitative study requires that the researcher and study participants view the study findings provide a credible, trustworthy, and authentic understanding of the phenomena (Yilmaz, 2013). The validity of the data collection

processes, analysis of the data and interpretation of the data help establish the trustworthiness and credibility of the study (Elo et al., 2014). Researchers must establish the credibility of their work, the dependability of the findings, the confirmability of the data and analysis, and the transferability of their research to demonstrate the trustworthiness of their research. (Hanson et al., 2011).

Dependability

A study establishes dependability by defining and explaining the research strategies, processes, and methods (Yilmaz, 2013). A study achieves dependability if it is repeatable with the same or comparable participants in an equivalent context (Lincoln & Guba, 1985). Establishing arduous sampling, data collection, data analysis, member checking, and other procedures increase the dependability of the study (Hanson et al., 2011). This study defines the eligibility of participants as software development leaders who have knowledge and experience using collaboration to minimize technical debt in software development. I used census sampling to ensure I collected data from all participants who were willing to participate from the study population.

Member checking is an important method for establishing dependability by allowing the participants to verify the accuracy of the researchers' account of their experiences (Lincoln & Guba, 1985; Onwuegbuzie & Byers, 2014). Researchers use member checking to establish the dependability of a study by discussing tentative themes and interpretations with participants (Hanson et al., 2011). I used member checking to increase the reliability of the study by confirming my interpretation of the data collected

with participants. I asked participants if my interpretations, descriptions, and themes accurately reflect their viewpoint.

Maintaining an audit trail of records, notes, and documents on all aspects of the research procedure enhance the dependability of a study (Cho & Lee, 2014). Explaining the research process, using triangulation, and describing the audit trail increases the dependability and reliability of the research (Zohrabi, 2013). Maintain records of the procedures, data collection process, data analysis steps, and development of interpretations allows other researchers to audit a study increasing the dependability of the study (Hanson et al., 2011). I clearly explained all the processes and phases of my research elaborating on every aspect of my study. I described in detail the purpose of the study, the design of the study, and the participants. I used triangulation by collecting and analyzing data from multiple sources to enhance the reliability of the results. I provided an audit trail by detailing the collection of data, analysis of the data, the development of my themes and interpretation of the results. The audit trail will facilitate others in replicating my research thus contributing to its dependability.

Credibility

A qualitative study is credible if the participants in the study found the results truthful (Yilmaz, 2013). A researcher establishes credibility, or internal validity, by using data triangulation, gathering rich descriptions, reaching data saturation and using an interview protocol (Hanson et al., 2011). The data collection procedures, data sources, triangulation, rich descriptions, and member checking affect the credibility of qualitative studies (Lincoln & Guba, 1985; Yilmaz, 2013). Member checking is the most important

technique for establishing credibility by allowing the participants to verify the accuracy and credibility of the researchers' account of their experiences (Lincoln & Guba, 1985). Researchers must collect data until redundancy of the data has occurred with no new information or themes being generated indicating saturation of the data (Walker, 2012). Data triangulation is a means to achieve data saturation, enhance the reliability of this study and ensure the validity of a study (Fusch & Ness, 2015; Lincoln & Guba, 1985). I used a data saturation grid to track emerging themes and data collected from interviews to determine when I had reached data saturation. I used member checking to ask participants if my rich descriptions and interpretations accurately reflect their experiences, meanings, and viewpoints.

Transferability

A determining factor in the transferability of a research study is the scope to which the findings are reproducible outside of the immediate research study or are generalizable to other contexts or settings (Lincoln & Guba 1985). The transferability, or external validity, of a study requires careful, detailed descriptions of the background, participants, sampling, population, and results of the research so others who read the research can determine if the results of the study will likely transfer to different situations with different participants (Hanson et al., 2011). A researcher can enhance the transferability of a study by providing thick descriptions of the study context and procedures allowing the readers to make decisions regarding transferability (Lincoln & Guba 1985). I have provided detailed descriptions of the background of my study, the

participant eligibility, the sampling methodology, and the population size of my study so other researchers may determine the transferability of my study to their setting.

Confirmability

The confirmability, or construct validity, of a study lies in the ability of others to review researchers' design, plan, and reasoning to ensure it makes sense (Hanson et al., 2011). The confirmability of a study depends on the ability of others to corroborate that the research took place using the methods and techniques the researcher describes (Lincoln & Guba 1985). A study experiences confirmability by using data to substantiate its findings, the interpretations are logical, and the results are clearly explained (Yilmaz, 2013). I recorded all procedures, data collection, analysis, and development of my interpretations enabling other researchers who did not conduct this study to review my plan and reasoning.

Methodological triangulation entails the use of multiple methods of data collection and analysis to develop a thorough understanding of the research phenomenon contributing to greater accuracy, reliability, and validity (Denzin, 1978; Hussein, 2015). Methodological triangulation benefits studies by providing richer data, correlating the data, increasing validity, and enhancing understandings of the research phenomena (Fusch & Ness, 2015). The within-method type of methodological triangulation uses complementary data collection and analysis methods to increase the accuracy and credibility of a study (Hussein, 2015). I used participant responses from interviews, interview observations, and organization documents as multiple methods of data collection. I used these multiple methods to enrich the data I collected by providing more

insight, a more comprehensive representation of the data and limiting inadequacies found in any one method. I used these multiple data collection methods to verify and validate my interpretations. I recognized any inconsistencies in the data and explained any unexpected findings using multiple methods.

Researchers use member checking to increase the reliability of a study by confirming their interpretations of the data collected with participants (Onwuegbuzie & Byers, 2014). Member checking is the process where researchers ask participants to confirm the researchers' interpretations of a participant's experiences, meanings, and viewpoints (Hanson et al., 2011). I requested all participants to confirm my interpretations of their interview responses to increase the reliability of my research.

Transition and Summary

In this section, I presented the details of my planned research study. The section included a description of the role of the researcher, participants, research design, sampling, data collection, and data analysis techniques. This section also included the procedures I employed to ensure the reliability and validity of my study. The next section includes my research findings, implications for social change, implications for practice, and recommendations for future research.

Section 3: Application to Professional Practice and Implications for Change

This study's focus was exploring the collaboration strategies that software development leaders use to reduce technical debt. In this section, I focus on the use of these findings in professional fields to bring about change. This section includes a study overview, presentation of findings, application to professional practice, implications for social change, recommendations for action, further study suggestions, personal reflections, and a conclusion.

Overview of Study

The purpose of this qualitative case study was to explore collaboration strategies software development leaders use to reduce the amount of technical debt created by software developers. The data came from interviews with software development leaders and organizational documentation from a large health care provider in California. The findings showed methods and tools that the software development leaders used to encourage collaboration, participation, and best practices to improve software quality and reduce technical debt.

Presentation of the Findings

This section contains a discussion of the four themes that emerged during the study. The purpose of the study was to answer the overarching research question: What collaboration strategies do software development leaders use to minimize technical debt created by their software developers? The answer to this question may be used to help solve the specific IT problem that some software development leaders lack collaboration strategies to reduce the amount of technical debt created by software developers. During

this study, I used semistructured interviews to collect data on the perceived usefulness and ease of use of collaboration strategies used by software development leaders. Additionally, I reviewed organizational documents related to software development standards, best practices, collaboration, and technical debt. Following the data collection and analysis, four main themes emerged: (a) extensive collaboration, (b) continuous verification, (c) participatory culture, and (d) tool support. These themes illustrate software development activities related to collaboration strategies.

Theme 1: Extensive Collaboration Is Critical

The first theme to emerge from data collection was that extensive collaboration within the development team is a critical component of the overall strategy of reducing technical debt. Participants reported that significant amounts of collaboration and peer reviews (see Table 1) at all stages of the development life cycle reduced the technical debt on their projects. Participant 8 asserted that development teams collaborate often on projects and was more concerned with too much collaboration rather than not enough collaboration. Participant 2 reported that collaboration on projects is the main reason why their projects do not accumulate technical debt. Participant 5 pointed out that the development teams follow agile methodologies that require frequent collaboration among team members. Participant 4 acknowledged that peer reviews are a large part of their overall collaboration strategy to reduce technical debt by clarifying misunderstandings and identifying design discrepancies, coding violations, and mistakes in projects. Participant 7 contended that developers' ability to successfully collaborate positively influences their job performance.

I found a similar emphasis on collaboration in the organizational documents I collected. My review of the organizational documents confirmed the extensiveness and importance of collaboration (see Table 1) within the organization to reduce technical debt. There were eight organizational documents referencing collaboration or collaborative activities such as status meetings, peer review meetings, and knowledge sharing. According to the *Agile Methodology Best Practices* document, all agile projects require frequent status, planning, review, and retrospective meetings throughout the life of the project. This aligned with interview data in which seven participants reported three or more types of status or peer review meetings conducted on projects. Participants 1, 3, 4, and 5 reported participating in status meetings, design reviews, code reviews, and project retrospective meetings. Participants 2, 4, 5, and 6 pointed out that the organization follows agile methodologies that are collaborative in nature.

There were five organizational documents containing checklists designed specifically for use during design reviews, code reviews, and quality assurance reviews to facilitate collaboration between developers and reviewers. Participants 2, 3, and 5 described using these checklist documents during peer review meetings to assess the quality of developers' work. The organizational document *Clarifying Roles and Responsibilities With RACI* facilitates the creation of a responsibility assignment matrix. This matrix identifies team members who are required to participate in collaboration activities during various tasks in a project. According to the document, a chart facilitates assigning team members the role of responsible, accountable, consulted, or informed for each activity in the project. Participant 8 reported that every project requires the creation

of a RACI chart to identify collaboration responsibilities of team members. Participants have found collaboration to be useful in reducing technical debt.

Table 1

Themes for Extensive Collaboration is Critical

Major/minor theme	Participant count	Document count
Extensive collaboration is critical	8	13
Status meetings	6	5
Design reviews	5	2
Code reviews	7	3
Retrospective meetings	8	2
Collaboration from methodology	4	8

The scholarly literature provided insight into the usefulness of collaboration in managing technical debt and aligned with the data from my interviews and organizational documents. Software developers collaborate extensively with team members to reduce technical debt. Ferzund et al. (2014) argued that software developers could spend more than 70% of their time collaborating with others leading to improved software development performance. Additionally, Shrivastava and Rathod (2017) found widespread collaboration on agile software development teams during requirement elicitation, coding, and testing. Shrivastava and Rathod found collaboration important for identifying and reducing technical debt. The findings of these two studies supported Participants 2, 4, and 5 who reported using agile methods with significant amounts of

collaboration during requirements, coding, and testing phases to minimize technical debt. These studies also aligned with the organizational documents I collected confirming the extensive use of collaboration. Caglayan and Bener (2016) indicated that the benefits from the number of collaborators might peak at some point, after which more collaborators might become a burden to the software developers. This supports Participant 8's concern of reaching a point where the team might be engaging in too much collaboration.

Software development teams could effectively reduce technical debt by establishing formal planning and review processes. Ozer and Vogel (2015) posited that software development organizations perform better when adopting formal rather than informal processes for sharing knowledge. Heikkila, Paasivaara, Lasssenius, Damian, and Engblom (2017) asserted that explicit planning strategies inherent to agile development might help prevent technical debt from accumulating. These studies aligned with the organizational documents I collected outlining agile best practices and formal checklists for peer review processes. Participants 1, 3, and 5 described agile planning processes and formal peer review meetings as means to reduce technical debt. Tom et al. (2013) contended that strong collaboration increases the visibility of technical debt making it easier for software development teams to identify and reduce technical debt. Behutiye, Rodríguez, Oivo, and Tosun (2017) found enhancing the visibility of technical debt through collaboration as one of the most significant strategies for managing technical debt. These studies supported Participants 1, 4, and 5 who indicated collaboration reduces

technical debt by improving the understanding, awareness, and visibility of team members.

I found participants believe collaboration is important to others, provides respect, improves standing, and is a mandatory requirement of their agile software development methodology. These findings are consistent with the conceptual framework TAM2 relating to social influences having a positive effect on perceived usefulness. Participant 4 indicated collaboration is a normal part of their software development practices and reported collaboration is important to the organization's leadership. Similarly, Participants 1 and 2 claimed collaboration was a mandatory part of their job. This aligned with Chan and Thong (2009) who concluded that social pressure and subjective norms are associated with teamwork and collaboration in software development. Likewise, Martinez et al. (2013) found that developers perceiving a development method as mandatory are more likely to adopt such methods. Participants 3 and 5 reported collaboration has a social component that gives respect to the people doing the collaboration and provides importance to others on the team. This aligned with Venkatesh and Davis (2000) who found in TAM2 that the importance of a person's social group, prestige, and standing in the organization significantly increases perceived usefulness. Subjective norms and image had a positive influence on participants' perceived usefulness of collaboration and their intention to use collaboration.

I found that participants believe collaboration during software development is a relevant and necessary part of their job. Participants 1, 2, and 5 reported collaboration is an essential part of their day-to-day activities. Additionally, Participant 3 indicated that

collaboration is key requirement of daily work. This aligned with Venkatesh and Davis (2000) who described the TAM2 construct job relevance as the perceived importance of a task to a person's job. Overhage and Schlauderer (2012) found developers perceive close, frequent collaboration to be a normal process in agile software development and relevant to their preferred work environment. These studies supported the participants' views that collaboration is a relevant part of their job in reducing technical debt.

I found participants believe collaboration supports their goal of producing quality software without technical debt. Participants 2 and 5 reported collaboration activities are the most useful job task in preventing technical debt. Additionally, Participants 7 and 8 indicated collaboration ensures the team follows best practices improving software quality and reducing technical debt. This aligned with Venkatesh and Davis (2000) who described the TAM2 construct output quality as the degree to which a task corresponds to a person's goals and how well the task performs. Additionally, Wallace and Sheetz (2014) used TAM2 to show software developers are more likely to perform development activities that increase the quality of the software they create. These studies supported the participants' views that collaboration supports their goal of producing quality software.

I found participants believe the results of their collaboration efforts are easily discernable and communicated within their team. Participants 2, 3, and 5 described using checklist documents during peer review meetings to identify and discuss the results of development work with others. Additionally, Participants 2, 3, 7, and 8 reported partaking in project retrospective meetings to discuss final project results with other team members. This aligned with Venkatesh and Davis (2000) who described the TAM2

construct result demonstrability as the degree to which use and positive results are easily discernable. Moreover, Riemenschneider et al. (2002) measured TAM2's result demonstrability by the ability of developers to easily ascertain results and communicate results to others. Similarly, Chan and Thong (2009) found the planning, development cycles, and frequent feedback in agile methodologies result in higher result demonstrability. These studies supported the participants' views that the results of collaboration efforts are easily discernable and effectively communicated with others.

Software development leaders should consider a development methodology that maximizes collaboration among their software development team members to reduce technical debt. Software development leaders should emphasize the social influences of collaboration by making it mandatory, promoting teamwork, and recognizing developers who perform well. The methodology should establish formal planning and review processes that include status meetings, design reviews, code reviews, and project retrospective meetings. These processes should clearly establish the objectives and importance of collaboration and align collaboration with the team's goals. Software development leaders should develop methods to verify the effectiveness of the collaboration activities.

Theme 2: Continuous Verification of Best Practices

Another theme that emerged from this study was that continuous verification of software development best practices is an important part of the collaboration strategies designed to reduce technical debt. Participants reported collaboration is the primary method they use to verify developers are following the organization's best practices (see

Table 2) to reduce technical debt. Participants 4 and 6 indicated the primary method of ensuring developers follow the organization's best practices is with collaboration. Participants 1, 2, 7, and 8 reported using peer reviews throughout the entire development process to reduce technical debt by verifying developers are following best practices. Participants 2, 3, 4, 5, and 8 indicated the use of checklists during peer review processes to verify developers are following the organization's development standards and best practices.

I found similar acknowledgments of continuous verification through peer reviews in the organizational documents. My review of the organizational documents confirmed the importance of verification processes (see Table 2) during collaboration to reduce technical debt. The *Code Review Checklist* document outlines a formal process for peer reviews consisting of a developer self-assessment, a reviewer secondary assessment, a team review meeting, and a final approval process. There were four similar documents providing guidelines for design reviews, test reviews, and performance reviews. These documents support Participants 1, 2, 4, and 5 who reported the verification processes provide structure to the development process. Additionally, Participants 2 and 8 reported peer reviews require a formal sign-off indicating approval of the review. These four checklist documents define the roles and responsibilities of team members in the peer review process, identify key metrics to assess, and require team members to assign a pass or fail to each metric. The organizational document, *Code Review Checklist,* requires team members to assess 142 different criteria spanning 17 major topics. These documents support Participants 2, 3, 4, 5, and 8 who reported the use of checklists during peer

review processes to verify developers are following development best practices. Researchers have also found combining collaboration and verification processes useful in reducing technical debt.

Table 2

Themes for Continuous Verification of Best Practices

Major/minor theme	Participant count	Document count
Continuous verification of best practices	8	13
Verifying best practices important	8	8
Verify designs using frameworks	5	2
Verify code using checklists	5	5
Verify completeness with sign offs	4	6
Verification provides structure	4	3

The usefulness of collaboration for verification and validation of best practices was an important theme in scholarly literature for managing technical debt that aligned with the data from my interviews and organizational documents. Inayat and Salim (2015) found software engineers use collaboration to facilitate discussions regarding best practices to improve software quality. Fairley and Willshire (2017) found that robust verification and validation processes could reduce technical debt by identifying violations of best practices during the development cycle. These studies support Participants 3, 4, and 5 who indicated the use of collaboration activities throughout the entire development process to verify developers are following best practices. These studies also align with

the organizational documents I collected confirming the importance of verification and validation processes during collaboration. Li, Liang, and Avgeriou (2015) asserted that violations of best practices cause architectural technical debt by compromising quality, maintainability, and evolvability. Tang and Lau (2014) found developers use design reviews to identify and correct potential design issues and use checklists to facilitate collaboration between architects, designers, and reviewers. These studies support the design guidelines and naming standards document that provide a checklist of 25 design best practices that require designers and reviewers verify before actual coding might begin. Additionally, Participant 5 claims that because all subsequent activities depend on design reviews, they are the most effective type of collaboration for preventing technical debt. Researchers have provided verification and validation strategies to reduce technical debt.

Software development teams could effectively reduce technical debt by establishing continuous verification processes using predefined checklists and designated roles for team members. Fitzgerald and Stol (2017) asserted continuous planning, verification, testing and other activities significantly improves the quality and resilience of software by detecting and fixing issues as soon as possible. Additionally, Fairley and Willshire (2017) found robust verification and validation processes reduce technical debt by identifying violations of best practices during the development cycle. These studies align with the participants who described continuous collaboration activities involving planning, verification, and testing as part of their overall strategy of reducing technical debt. Fitzgerald and Stol found peer reviews using predefined checklists were more

effective in achieving quality than reviews performed without a checklist. Additionally, Lopez-Martín, Nassif, and Abran (2017) found design review and code review checklists provide a framework to software development projects focusing on finding defects at earlier stages in the development cycle. These studies support the organizational documents I collected outlining formal checklists for peer review processes. Participants 2, 3, 4, 5, and 8 reported checklists were an important part of their peer review processes to verify best practices.

The conceptual framework provided insight into the participants' perceived usefulness of verification processes. I found participants believe verification and validation processes are important to the team and a mandatory part of the development process. Participants 2 and 8 reported checklists are a mandatory part of the development process requiring a formal sign off. Additionally, the checklist documents I reviewed also require a formal approval indicating they are mandatory. This aligns with Venkatesh and Davis (2000) who described TAM2 construct subjective norm has a positive effect on perceived usefulness. Moreover, Riemenschneider et al. (2002) found that developers perceiving a development process as mandatory are more likely to adopt such methods. Participant 3 reported the entire team reviews checklists so it is important they are fully completed. Furthermore, Participant 4 reported an expectation that everyone must go through the best practices checklists. This aligns with Venkatesh and Davis who found in TAM2 that the importance to a person's social group significantly increases perceived usefulness. Additionally, Nel et al. (2016) found software developers were more likely to use verification processes if they perceive the processes as important to others. Subjective

norm had a positive influence on participants' perceived usefulness of collaboration and their intention to use collaboration.

I found that participants believe verification processes are relevant to their job, improves software quality, and provides tangible results. These findings are consistent with Venkatesh and Davis (2000) who theorized in TAM2 that job relevance, output quality, and result demonstrability influence perceived usefulness and intention to use. Participant 4 indicated verification processes and checklists were an important part of their software development methodology and that they expect everyone to be part of the process. This aligns with Overhage and Schlauderer (2012) who found developers were more likely to use a development process if they found it compatible with their job. Participant 2 reported using checklists during peer reviews improves the quality of the software they deliver. This corresponds to Wallace and Sheetz (2014) who examined output quality in TAM2 and found software developers are more likely to perform a software verification processes that increase the quality of the software they create. The organizational document, *Code Review Checklist,* requires developers and reviewers to assess 142 different criteria and share the results with the entire team. This aligns to Riemenschneider et al. (2002) who measured result demonstrability of TAM2 by the ability of developers to easily ascertain results, explain results to others, and easily communicate results to others. These studies support the participants' views that verification processes and checklists are pertinent development activities, improve software quality, and provide tangible results.

Software development leaders should establish continuous verification processes as a mandatory part of their software development processes. The verification processes should use predefined checklists that align software quality measures with development best practices. Software development leaders should clearly define the roles, responsibilities, and expectations of developers and reviewers during verification processes. The verification processes should be transparent and the results shared with the entire development team. Continuous verification processes are an important part of the overall collaboration strategies for reducing technical debt in an organization.

Theme 3: Participatory Culture Improves Clarity and Collectiveness

The third theme that emerged from this study was establishing a participatory culture reduces technical debt by improving clarity, awareness, and collectiveness in software development. Participants reported a collaborative culture that promotes participation, understanding, awareness, and collectiveness (see Table 3) helping to reduce technical debt. Participants 3 and 5 reported onsite, offsite and offshore team members are all included in peer reviews and are equally involved in the collaboration. Additionally, Participants 3, 4, and 5 indicated getting everyone's point of view improves understanding, identifies potential issues, and lowers project risk. In addition, during my tour of the organization's site, I observed that the team referred to large conference rooms as collaboration rooms, which aligned with the participatory nature of the team. Participants 2 and 5 reported a social component to collaboration that improves the team performance by enhancing relationships and awareness of teammates. Additionally, Participants 1 and 5 reported face-to-face collaboration is preferred because it fosters

participation, knowledge sharing, and understanding among the team. Moreover, Participants 1 and 2 indicated their culture of collaboration improves problem-solving, defect detection, and software quality, which ultimately reduce technical debt. The organizational documents I collected failed to consider the role of participatory culture in collaboration. Researchers have also found the relationship between participatory culture and collaboration in software development teams was useful in reducing technical debt.

Table 3

Themes for Participatory Culture Improves Clarity and Collectiveness

Major/minor theme	Participant count
Participatory culture improves clarity and collectiveness	6
Participation encouraged and promoted	4
Direct communication is important	5
Environment supports collectiveness	3
Culture improves clarity	5

The impact of software development culture on collaboration and software quality was an important theme in the scholarly literature. Storey, Zagalsky, Filho, Singer, German (2017) contend a participatory culture is one that lowers barriers to participation, supports community building, facilitates mentoring, and values subjective norm. Additionally, Santos, Goldman, and de Souza (2015) found the adaptive capacity of the organizational environment affects the effectiveness of inter team knowledge sharing. These studies support participants' reports of a culture that promotes participation,

awareness, sharing, and collectiveness. Storey et al. found most developers find face-to-face communication the best method to discuss ideas, provide explanations, and avoid misunderstandings. Moreover, Femmer et al. (2016) found fewer ambiguities during software development positively affects software quality reducing technical debt. These studies align with Participants 1 and 5 who reported face-to-face collaboration allowed them to clarify their understandings and avoid uncertainties in the development process. Carver et al. (2015) found the number of collaborators during a peer review positively correlates to software quality. Similarly, McIntosh, Kamei et al. (2015) found the higher levels of participation in peer reviews results in lower defect rates. These studies confirm reports from Participants 3 and 5 that high levels of participation in collaboration activities provide multiple points of view. A study by Rola, Kuchta, Kopczyk (2016) found open working environments and shared office spaces significantly improve collaboration. This supports Participants 1 and 2 who reported teams occupied large, open rooms to facilitate collaboration. The conceptual framework provided insight into the impact of a participatory culture on job performance.

I found that participants believe a participatory culture improves their job performance and reduces technical debt. Participants 3 and 5 reported high levels of participation in collaboration activities improves the quality of their work. This aligns with Cheung and Vogel (2013) who found user participation in collaboration is an indicator of TAM2's perceived usefulness of collaboration in enhancing one's job performance. Participants 3, 4, and 5 indicated collaboration improves understanding, awareness, and perceptions. This corresponds with Thakurta and Roy (2012) who posited

project uncertainty is the antecedent of result demonstrability defined by TAM2. They found clarity, awareness, and visibility on development projects positively influences the perceived usefulness of the activities. Participants 2 and 5 reported that collaboration contains a social component that improved team cohesion. This matches Overhage and Schlauderer (2012) who found developer perceptions of team cohesiveness derived from close collaboration with other developers influences TAM2's output quality. Participants 1 and 2 reported their single, shared office environment encourages collaboration and knowledge sharing within the team. This corresponds to Thakurta and Roy who found a project environment encouraging participation influences TAM2's perceived usefulness and ease of use of development activities.

Software development leaders should establish a software development culture that lowers barriers to participation, supports team building, facilitates mentoring, and fosters collaboration. Leaders should promote an adaptive, open working environment that promotes direct communication, awareness, and collectiveness. Encouraging a participatory culture will improve software quality and reduce technical debt by avoiding uncertainty in development projects, obtaining multiple points of view, and increasing team cohesion. A participatory culture and open environment are an important part of the overall collaboration strategies for reducing technical debt in an organization.

Theme 4: Collaborative Tools Support Distributed Teams

The final theme that emerged from this study was that collaborative tools are necessary to support awareness, knowledge sharing, and participation in distributed software development teams. Participants reported WebEx, Skype, and SharePoint (see

Table 4) promoted participation, awareness, and knowledge sharing with distributed team members helping to reduce the risk of technical debt. WebEx is a tool primarily used for online meetings and video conferencing. Skype is a tool primarily used for instant messaging, but also supports online meetings, screen sharing, and video conferencing. SharePoint is a collaborative platform for sharing files and information. Participants 4, 5, and 7 reported using WebEx to extend participation in meetings, peer reviews, and training sessions to distributed team members. Additionally, Participants 1 and 2 reported that the interactive nature of online meetings makes it easier to ask questions and verify team members' understandings. Participants 3 and 8 reported instant messaging was the fastest and most efficient method of exchanging knowledge, asking questions, and clarifying interpretations. Additionally, Participant 1 reported using instant messaging to facilitate group discussions within the team. Participants 1 and 5 indicated that SharePoint is a knowledge-sharing repository to store project specific documents and documents related to software development best practices. All participants indicated the use of email for collaborating with remote team members was an important method for collaboration. However, Participants 1, 2, 4, and 5 indicated that they use email only when a person was not available using instant messaging or online meetings.

My review of the organizational documents confirmed the importance of tools (see Table 4) for collaborating with distributed team members. The *Agile Software Development Methods* document indicated the use of video conferencing and instant messaging tools for use with geographically dispersed teams. This corroborates the seven participants who reported the use of WebEx or Skype for video conferencing and instant

messaging. The organizational document, *Tableau Drive Manual*, recommends collaborative capacity building using virtual meetings. This aligns with Participants 4, 5, and 7 who reported using virtual online meetings to extend participation to distributed team members. The *Agile Kanban Methods* document specified the use of Jira as a tool to share planning, tracking, and status information of team members. This supports Participants 2 and 5 who reported using Jira to share awareness of team member activities. There were three organizational documents containing links to other documents located in SharePoint. This validates Participants 1 and 5 who indicated they use SharePoint as a document and knowledge-sharing repository. Researchers have also found collaborative tools useful in reducing technical debt in distributed development environments.

Table 4

Themes for Collaborative Tools Support Distributed Teams

Major/minor theme	Participant count	document Count
Collaborative tools support distributed teams	8	6
Online meetings (WebEx)	7	2
Instant messaging (Skype)	7	1
Email (Outlook)	8	0
Document sharing (SharePoint, Jira, Box)	5	4

The benefits of collaborative tools in distributed development teams was an important theme in the scholarly literature. Khan and Khan (2017) found that using

WebEx and Skype could improve team cohesion in geographically dispersed teams. This aligns with Participants 4, 5, and 7 who reported using WebEx to include offshore team members in meetings. Additionally, Khan and Khan found distributed development teams could improve knowledge sharing, awareness, and understandings by using online meetings and instant messaging. This supports Participants 1 and 2 who reported online meetings make it easier to share knowledge and verify understandings. Giuffrida and Dittrich (2013) found instant messaging improves awareness, facilitates knowledge sharing, and mostly consists of asking questions. This aligns with Participants 3 and 8 who reported instant messaging was the most efficient method of exchanging knowledge and asking questions. Furthermore, Giuffrida and Dittrich found instant messaging leads to spontaneous, informal collaboration that is faster than email. This supports Participants 1, 2, 4, and 5 who indicated they use email only when a person was not available using instant messaging. The conceptual framework provided insight into the perceived usefulness and ease of use of collaborative tools in software development.

I found that participants' perceived usefulness and ease of use of the collaborative tools determine their intention to use the tools. Participants 1 and 2 reported WebEx made it very easy to ask questions and verify understandings from any device. Participant 3 reported instant messaging was the easiest method of collaboration. This aligns with Cheung and Vogel (2013) who found TAM2's perceived ease of use significantly influences the use of collaborative tools. Additionally, Park et al. (2014) found the perceived ease of use of a teleconferencing system significantly influences the perceived usefulness of the system. Participants 4 and 6 reported WebEx was very useful for

sharing knowledge, clarifying understandings, and using in team-based activities. This is supported by Maruping and Magni (2015) who found the TAM2 construct perceived usefulness could be measured by IT professionals' belief that collaboration technology would improve knowledge sharing, teamwork, and accessibility. Moreover, Godin et al. (2017) found the perceived usefulness of WebEx to accomplish future work significantly influences virtual team members' intention to use WebEx. Participants 4, 5, and 7 reported collaborative tools were useful in fostering participation among offshore team members. This aligns with Cheung and Vogel who found the TAM2 construct subjective norm significantly influences the use of collaborative tools especially when team members believe participation is important.

Applications to Professional Practice

This study identified collaboration strategies software development leaders could apply to their development organizations to reduce the technical debt accumulated by their developers. Software development leaders should consider a development methodology that maximizes collaboration within their software development team to reduce technical debt. Agile, Lean, Scrum and Kanban are examples of software development methodologies the promote collaboration. The methodology should establish formal planning and review processes that include status meetings, design reviews, code reviews, and project retrospective meetings. Additionally, these processes should establish objectives, identify quality metrics, highlight the importance of collaboration, and align collaboration with the team's goals. Software development leaders need to ensure developers understand the usefulness and relevance of

collaboration for reducing technical debt and improving quality. Software development leaders should emphasize the social influences of collaboration by making it mandatory, promoting teamwork, and recognizing developers who perform well. Software development leaders should develop methods to verify the effectiveness of the collaboration activities.

Software development leaders should establish continuous verification methods as a mandatory part of their software development processes. The verification processes should use predefined checklists that align software quality measures with development best practices. Sharing of all verification results with the team is important so the developers can see the tangible results of their efforts. Software development leaders should define the roles, responsibilities, and expectations of developers and reviewers during verification processes. Verification processes will only be successful if developers understand their obligations. The verification processes should be transparent and the results shared with the entire development team. Continuous verification processes are an important part of the overall collaboration strategies for reducing technical debt in an organization. Developers will be more willing to participate in verification processes if they understand the importance, view the processes as mandatory, and see quantifiable results.

Software development leaders should establish a software development culture that lowers barriers to participation, supports teamwork, facilitates mentoring, and fosters collaboration. Collaboration is dependent upon participation and cannot exist without it. Teamwork and mentoring allow developers to solve complex problems and learn from

each other. Leaders should promote an adaptive, open working environment that promotes direct communication, awareness, and collectiveness. Direct communication is fast, efficient, and reduces potential misunderstandings between development staff. Improved awareness allows developers to identify areas where they might be able to provide knowledge or help to others. Encouraging a participatory culture will improve software quality and reduce technical debt by avoiding uncertainty in development projects, obtaining multiple points of view, and increasing team cohesion. A participatory culture and open environment are an important part of the overall collaboration strategies for reducing technical debt in an organization.

Software development leaders should ensure developers have a variety of collaboration tools available for their use. In addition to the normal email and document sharing software, leaders should provide video conferencing and instant messaging software. Video conferencing software such as WebEx will allow geographically dispersed team members to participate in peer review and meetings. Instant messaging software such as Skype will allow developers to partake in spontaneous, informal collaboration that is more efficient than email. These collaborative tools will improve knowledge sharing, awareness, teamwork, and accessibility. Software development leaders should seek recommendations from developers on which tools they perceive the most useful in their daily activities and the easiest to use. The availability of collaborative tools is an important part of the overall collaboration strategies for reducing technical debt.

Implications for Social Change

This study explored how collaboration among diverse individuals can benefit a common goal of reducing technical debt by improving software quality. The benefits of collaboration and software quality are not limited to the single organization in my study. Collaboration extends far beyond software development and reaches every area of society. Collaboration brings people of diverse backgrounds, different perspectives, and varying skill sets together to achieve a common goal. This includes sharing knowledge and working together to solve a common problem. A society only exists if people work together. During collaboration activities, people improve their communication skills by improving their ability to express themselves and interpret the communications of others. Additionally, collaboration teaches people how to build relationships, establish trust, and respect the ideas and opinions of others. The knowledge learned from this study can provide positive social change both inside and outside of the workplace. A society cannot exist without collaboration and society will benefit from improving peoples' ability to work together and fostering respect for one another.

The benefits of improved software quality also extend beyond the organization in this study. Society has become increasingly dependent on computer software for most daily tasks. Computer software controls transportation, energy, food and medical care, which are vital components of any society. Poor software quality can disrupt society, cause hardships, and even lead to death. A study by Wong, Li, Laplante, and Siok (2017) found several instances of poor software quality that had severe adverse effects on society. In 1992, technical debt in the London Ambulance Service's new computer

dispatch system disrupted medical care that may have led to the deaths of 20 people (Wong et al., 2017). Technical debt in an energy management system of an Ohio power company caused a blackout in 2003 that affected over 50 million people and cost society $13 billion (Wong et al., 2017). The knowledge learned from this study can help improve the quality of software society depends on every day by preventing hardships and making peoples' lives better. As society's dependence on computer software grows, this knowledge will become more important.

Recommendations for Action

I explored the collaboration strategies used by an organization to reduce technical debt created by software developers. The study findings showed an environment that promotes collaboration, a culture that encourages participation, and accessibility to collaborative tools successfully reduced technical debt in the case organization. Software development leaders should adopt Agile based methodologies that emphasize collaboration throughout the development life cycle. Additionally, leaders should continuously verify adherence to best practices by requiring formal peer review processes at all phases of development projects. It is essential for software development leaders to define roles, responsibilities, and expectations of developers during peer reviews. Collaboration is more likely to occur if it is a mandatory part of the daily development activities.

Software development leaders should explore techniques to stimulate participation and teamwork in collaboration activities. One technique could be to require all developers participate in design and code reviews to improve clarity, awareness, and

visibility in projects. Another technique could be to assign pairs of developers to conduct code reviews to improve teamwork. Moreover, pairing a senior and junior developer during code reviews would allow the senior developer to mentor the junior developer. Software development leaders should situate developers in large, open workspaces to foster face-to-face collaboration. The availability of additional private meeting rooms for team activities will minimize disruptions in the open workspaces.

The CIOs of organizations with multiple software development teams should ensure the software development leaders are sharing collaboration strategies and results with each other. Furthermore, CIOs should work with software development leaders to ensure developers have a variety of collaboration tools available for their use. These tools should include video conferencing and instant messaging software to improve participation of dispersed team members.

In general, this study might be beneficial to key community stakeholders, software development leaders, and the software development community. I will disseminate a high-level summary of the results of this study to the community stakeholders and research participants via email. Wherever possible, I intend to share the research results using effective and appropriate platforms such as my place of employment, lectures, conferences, trade journals, and training seminars.

Recommendations for Further Study

Several limitations of this study warrant further research. The research methods of this study imposed limitations on the results due to the chosen design, participants, organization, data collection, and other aspects of the study. The first limitation of this

study was the subjective nature of qualitative studies that might inject bias into the study. Additionally, this study was limited to a single health care organization in the state of California. I recommend additional qualitative research studies that include other organizations, industries, and locations to see if the findings from new studies correspond to my findings. The study findings were restricted to software development leaders due to the narrow participant criteria of the study population. I recommend additional qualitative studies explore the perceptions of software developers, database administrators, quality assurance personnel, and others involved in software development projects. Another limitation was that the data collection was limited to interview questions and organizational documents. I recommend additional qualitative studies with expanded data collection to include focus groups and observations from peer reviews and other meetings. Lastly, this study was limited to a single case organization restricting the generalizability of the results outside of the case organization. I recommend a quantitative study to determine if the results of this study are generalizability outside of the single case organization.

The study findings also identified areas that merit additional research. The study findings highlighted the significance of collaboration in peer review meetings but did not explore the actual procedures or observe actual meetings. Additionally, the use of checklists in these meetings was also significant but the study did not explore the contents of the checklists. I recommend further research explore the impact of peer review techniques on collaboration and technical debt. One unexpected comment from a participant was the notion that a development team could have too much collaboration. I

recommend further research explore the amount of collaboration on development teams to identify if and when it incurs negative consequences.

Reflections

The doctoral study process was a journey filled with obstacles, but also enlightenment. Each time I encountered an obstacle it was difficult, but I persevered and expanded my knowledge in the process. During this doctoral study, I learned how to conduct academic research, how to analyze research, and how research affects others. During my journey, I found that I really enjoy learning, conducting research, and discussing research with others.

Having been involved in software development for over two decades, I understand the role of collaboration in projects and the benefits of minimizing technical debt. I took painstaking efforts to remain objective during the study and prevent any personal bias or preconceived notions from affecting the results. I acknowledged potential biases throughout the research study when possible. Due to the semistructured nature of the interview questions, it is possible that I unintentionally biased the research through interactions with participants. I did my best to ensure the reliability and credibility of this study. I believe I learned a lot from the participants in regards to the social aspects of collaboration in software development.

Summary and Study Conclusions

Software development teams require a significant amount of collaboration to produce high-quality software with minimal technical debt. Collaboration requires an environment that encourages developer interactions, a culture that promotes developer

participation, and a set of tools that facilitate teamwork. Continuous verification processes are the primary methods of collaboration to ensure developers follow best practices and manage technical debt.

Cognitive and social processes effect the usefulness of collaboration. Software development leaders should integrate collaborative activities into daily activities, develop metrics to measure technical debt, and publicize the results. Additionally, leaders should emphasize the importance of collaboration by making it mandatory, promoting teamwork, and recognizing developers who perform well.

References

Abdullah, F., & Ward, R. (2016). Developing a general extended technology acceptance model for e-learning (GETAMEL) by analysing commonly used external factors. *Computers in Human Behavior, 56*, 238–256. doi:10.1016/j.chb.2015.11.036

Abma, T. A., & Stake, R. E. (2014). Science of the particular: An advocacy of naturalistic case study in health research. *Qualitative Health Research, 24*(8), 1150–1161. doi:10.1177/1049732314543196

Aliyu, A. A., Bello, M. U., Kasim, R., & Martin, D. (2014). Positivist and non-positivist paradigm in social science research: Conflicting paradigms or perfect partners? *Journal of Management and Sustainability, 4*(3), 79. doi:10.5539/jms.v4n3p79

Alves, N. S. R., Mendes, T. S., de Mendonca, M. G., Spínola, R. O., Shull, F., & Seaman, C. (2016). Identification and management of technical debt: A systematic mapping study. *Information and Software Technology, 70*, 100–121. doi:10.1016/j.infsof.2015.10.008

Ampatzoglou, A., Ampatzoglou, A., Chatzigeorgiou, A., & Avgeriou, P. (2015). The financial aspect of managing technical debt: A systematic literature review. *Information and Software Technology, 64*, 52–73. doi:10.1016/j.infsof.2015.04.001

Anfara, V. A., Brown, K. M., & Mangione, T. L. (2002). Qualitative analysis on stage: Making the research process more public. *Educational Researcher, 31*(7), 28–38. doi:10.3102/0013189X031007028

Anney, V. N. (2014). Ensuring the quality of the findings of qualitative research:

Looking at trustworthiness criteria. *Journal of Emerging Trends in Educational Research and Policy Studies, 5*(2), 272–281. Retrieved from http://jeteraps.scholarlinkresearch.com.

Astalin, P. K. (2013). Qualitative research designs: A conceptual framework. *International Journal of Social Science & Interdisciplinary Research, 2*(1). Retrieved from http://www.i-scholar.in/index.php/ijssir.

Avgeriou, P., Kruchten, P., Nord, R. L., Ozkaya, I., & Seaman, C. (2016). Reducing friction in software development. *IEEE Software, 33*(1), 66–73. doi:10.1109/MS.2016.13

Bacchelli, A., & Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. In *Software Engineering (ICSE), 2013 35th International Conference on* (pp. 712–721). IEEE. doi:10.1109/ICSE.2013.6606617

Barczak, G. (2015). Publishing qualitative versus quantitative research. *Journal of Product Innovation Management, 32*(5), 658. doi:10.1111/jpim.12277

Baysal, O., Kononenko, O., Holmes, R., & Godfrey, M. W. (2015). Investigating technical and non-technical factors influencing modern code review. *Empirical Software Engineering*, 1–28. doi:10.1007/s10664-015-9366-8

Behutiye, W. N., Rodríguez, P., Oivo, M., & Tosun, A. (2017). Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Information and Software Technology, 82*, 139–158. doi:10.1016/j.infsof.2016.10.004

Bell, M., Charles-Edwards, E., Kupiszewska, D., Kupiszewski, M., Stillwell, J., & Zhu,

Y. (2015). Internal migration data around the world: Assessing contemporary practice. *Population, Space and Place, 21*(1), 1–17. doi:10.1002/psp.1848

Bengtsson, M. (2016). How to plan and perform a qualitative study using content analysis. *NursingPlus Open, 2*, 8–14. doi:10.1016/j.npls.2016.01.001

Bernauer, J. A., Lichtman, M., Jacobs, C., & Robinson, S. (2013). Blending the old and the new: Qualitative data analysis as critical thinking and using Nvivo with a generic approach. *Qualitative Report, 18*(31), 1–10. Retrieved from http://nsuworks.nova.edu/tqr/.

Beskow, L. M., Check, D. K., & Ammarell, N. (2014). Research participants' understanding of and reactions to certificates of confidentiality. *AJOB Empirical Bioethics, 5*(1), 12–22. doi:10.1080/21507716.2013.813596.

Biederer, M., Arguel, A., Liu, J., & Lau, A. (2014). From web-based to mobile: Experiences of developing a personally controlled health management system. *Health Informatics Society of Australia, 8*(1). Retrieved from https://www.hisa.org.au/.

Birt, L., Scott, S., Cavers, D., Campbell, C., & Walter, F. (2016). Member checking: A tool to enhance trustworthiness or merely a nod to validation? *Qualitative Health Research, 26*(13), 1802–1811. doi:10.1177/1049732316654870

Boblin, S. L., Ireland, S., Kirkpatrick, H., & Robertson, K. (2013). Using Stake's qualitative case study approach to explore implementation of evidence-based practice. *Qualitative Health Research, 23*(9), 1267–1275. doi:10.1177/1049732313502128

Breed, B., Mentz, E., & van der Westhuizen, G. (2014). A metacognitive approach to pair programming: Influence on metacognitive awareness. *Electronic Journal of Research in Educational Psychology, 12*(1), 33–60. doi:10.14204/ejrep.32.13104

Caglayan, B., & Bener, A. B. (2016). Effect of developer collaboration activity on software quality in two large scale projects. *Journal of Systems and Software*. doi:10.1016/j.jss.2016.03.055

Caine, V., Estefan, A., & Clandinin, D. J. (2013). A return to methodological commitment: Reflections on narrative inquiry. *Scandinavian Journal of Educational Research, 57*(6), 574–586. doi:10.1080/00313831.2013.798833

Carver, J. C., Caglayan, B., Habayeb, M., Penzenstadler, B., & Yamashita, A. (2015). Collaborations and code reviews. *IEEE Software, 32*(5), 27–29. doi:10.1109/MS.2015.113

Castillo-Montoya, M. (2016). Preparing for interview research: The interview protocol refinement framework. *Qualitative Report, 21*(5), 811–831. Retrieved from http://nsuworks.nova.edu/tqr/.

Castleberry, A. (2014). NVivo 10 [software program]. Version 10. QSR International; 2012. *American Journal of Pharmaceutical Education, 78*(1). doi:10.5688/ajpe78125

Chan, F. K. Y., & Thong, J. Y. L. (2009). Acceptance of agile methodologies: A critical review and conceptual framework. *Decision Support Systems, 46*(4), 803–814. doi:10.1016/j.dss.2008.11.009

Cheung, R., & Vogel, D. (2013). Predicting user acceptance of collaborative

technologies: An extension of the technology acceptance model for e-learning. *Computers & Education, 63*(0), 160–175. doi:10.1016/j.compedu.2012.12.003

Cho, J. Y., & Lee, E.-H. (2014). Reducing confusion about grounded theory and qualitative content analysis: Similarities and differences. *Qualitative Report, 19*(32), 1–20. Retrieved from http://nsuworks.nova.edu/tqr/.

Codabux, Z., & Williams, B. (2013). Managing technical debt: An industrial case study. In *Managing Technical Debt (MTD), 2013 4th International Workshop on* (pp. 8–15). IEEE. doi:10.1109/MTD.2013.6608672

Coman, I. D., Robillard, P. N., Sillitti, A., & Succi, G. (2014). Cooperation, collaboration and pair-programming: Field studies on backup behavior. *Journal of Systems and Software, 91*, 124–134. doi:10.1016/j.jss.2013.12.037

Congyingzi, Z., & Yan, W. (2016). A flowchart for rapid technical debt management decision making. *Journal of Software, 11*(2), 212–219. doi:10.17706/jsw.11.2.212-219

Conrad, E. D. (2013). Willingness to use strategic IT innovations at the individual level: An empirical study synthesizing DOI and TAM theories. *Academy of Information & Management Sciences Journal, 16*(1), 99–110. Retrieved from http://www.alliedacademies.org/.

Cook, A. F., Hoas, H., & Joyner, J. C. (2013). The Protectors and the Protected: What Regulators and Researchers Can Learn from IRB Members and Subjects. *Narrative Inquiry in Bioethics, 3*(1), 51–65. doi:10.1353/nib.2013.0014

Cronin, C. (2014). Using case study research as a rigorous form of inquiry. *Nurse*

Researcher, 21(5), 19–27. doi:10.7748/nr.21.5.19.e1240

Crowhurst, I. (2013). The fallacy of the instrumental gate? Contextualising the process of gaining access through gatekeepers. *International Journal of Social Research Methodology, 16*(6), 463–475. doi:10.1080/13645579.2013.823282

Cruz, E.V., & Higginbottom, G. (2013). The use of focused ethnography in nursing research. *Nurse Researcher, 20*(4), 36–43 8p. doi:10.7748/nr2013.03.20.4.36.e305

Cubric, M. (2013). An agile method for teaching agile in business schools. *The International Journal of Management Education, 11*(3), 119–131. doi:10.1016/j.ijme.2013.10.001

Cunningham, W. (1992). The WyCash portfolio management system. In *ACM SIGPLAN OOPS Messenger* (Vol. 4, pp. 29–30). doi:10.1145/157709.157715

Curtis, B., Sappidi, J., & Szynkarski, A. (2012). Estimating the principal of an application's technical debt. *IEEE Software, 29*(6), 34–42. doi:10.1109/MS.2012.156

Davis, F. D., Bagozzi, R. P., & Warshaw, P. R. (1989). User acceptance of computer technology: A comparison of two theoretical models. *Management Science, 35*(8), 982–1003. doi:10.1287/mnsc.35.8.982

Davis Jr, F. D. (1986). *A technology acceptance model for empirically testing new end-user information systems: Theory and results* (Doctoral dissertation). Massachusetts Institute of Technology. Retrieved from http://hdl.handle.net/1721.1/15192.

De Massis, A., & Kotlar, J. (2014). The case study method in family business research: Guidelines for qualitative scholarship. *Journal of Family Business Strategy, 5*(1), 15–29. doi:10.1016/j.jfbs.2014.01.007

DeFeo, D.J. (2013). Toward a model of purposeful participant inclusion: Examining deselection as a participant risk. *Qualitative Research Journal, 13*(3), 253–264. doi:10.1108/qrj-01-2013-0007

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

Denzin, N. K. (1978). *Sociological methods*. New York, NY: McGraw-Hill.

di Bella, E., Fronza, I., Phaphoom, N., Sillitti, A., Succi, G., & Vlasenko, J. (2013). Pair programming and software defects-A large, industrial case study. *IEEE Transactions on Software Engineering, 39*(7), 930–953. doi:10.1109/TSE.2012.68

di Russo, D., & Douglas, M. (2013). The validity of the technology acceptance model in collaboration system software. *Business and Management Review, 3*, 1–5. Retrieved from http://www.bmr.businessjournalz.org/.

Dikko, M. (2016). Establishing construct validity and reliability: Pilot testing of a qualitative interview for research in Takaful (Islamic Insurance). *Qualitative Report, 21*(3), 521–528. Retrieved from http://nsuworks.nova.edu/tqr/.

Donaldson, D. R. (2016). The digitized archival document trustworthiness scale. *International Journal of Digital Curation, 11*(1), 252–270. doi:10.2218/ijdc.v11i1.387

Doody, O., & Noonan, M. (2013). Preparing and conducting interviews to collect data. *Nurse Researcher, 20*(5), 28–32 5p. doi:10.7748/nr2013.05.20.5.28.e327

Dullemond, K., Van Gameren, B., & Van Solingen, R. (2014). Collaboration spaces for virtual software teams. *Software, IEEE, 31*(6), 47–53. doi:10.1109/MS.2014.105

Dunne, B., Pettigrew, J., & Robinson, K. (2016). Using historical documentary methods to explore the history of occupational therapy. *British Journal of Occupational Therapy, 79*(6), 376–384. doi:10.1177/0308022615608639

Edwards-Jones, A. (2014). Qualitative data analysis with NVIVO. *Journal of Education for Teaching, 40*(2), 193–195. doi:10.1080/02607476.2013.866724

Eliasson, U., Martini, A., Kaufmann, R., & Odeh, S. (2015). Identifying and visualizing architectural debt and its efficiency interest in the automotive domain: A case study. In *Managing Technical Debt (MTD), 2015 IEEE 7th International Workshop on* (pp. 33–40). IEEE. doi:10.1109/MTD.2015.7332622

Elo, S., Kaariainen, M., Kanste, O., Polkki, T., Utriainen, K., & Kyngas, H. (2014). Qualitative Content Analysis. *SAGE Open, 4*(1). doi:10.1177/2158244014522633

Eservel, U. Y. (2014). IT-Enabled knowledge creation for open innovation. *Journal of the Association for Information Systems, 15*(11), 805–834. Retrieved from http://aisel.aisnet.org/jais/.

Esichaikul, V., Win, M. A., Bechter, C., & Rehman, M. (2013). Development and evaluation of wiki collaboration space for e-learning. *Journal of Enterprise Information Management, 26*(5), 536–552. doi:10.1108/jeim-07-2013-0045

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), 1–4. doi:10.11648/j.ajtas.20160501.11

Evans, R. D., Gao, J. X., Martin, N., & Simmonds, C. (2015). Integrating social knowledge and collaboration tools into dispersed product development. *International Journal of Advanced Corporate Learning, 8*(2), 20–27. doi:10.3991/ijac.v8i2.4548

Fagan, M. (1976). Design and Code Inspections to Reduce Errors in Program Development. *IBM Systems Journal, 15*(3). doi:10.1147/sj.153.0182

Fairley, R. E., & Willshire, M. J. (2017). Better now than later: Managing technical debt in systems development. *Computer, 50*(5), 80–87. doi:10.1109/MC.2017.124

Femmer, H., Fernandez, D. M., Wagner, S., & Eder, S. (2016). Rapid quality assurance with requirements smells. *Journal of Systems and Software.* doi:10.1016/j.jss.2016.02.047

Ferreria, R., Buttell, F., & Ferreria, S. (2015). Ethical considerations for conducting disaster research with vulnerable populations. *Journal of Social Work Values and Ethics, 12*(1), 379–384. Retrieved from http://jswve.org/.

Ferzund, J., Yasrab, R., & Razzaq, S. (2014). Web 2.0 and collaborative software development. *International Journal of Software Engineering and Its Applications, 8*(7), 107–120. doi:10.14257/ijseia.2014.8.7,09

Fetters, M. D., Curry, L. A., & Creswell, J. W. (2013). Achieving integration in mixed methods designs - Principles and practices. *Health Services Research, 48*(6pt2), 2134–2156. doi:10.1111/1475-6773.12117

Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software, 123*, 176–189. doi:10.1016/j.jss.2015.06.063

Fletcher, J., Sarkani, S., & Mazzuchi, T. A. (2014). A technology adoption model for broadband Internet adoption in India. *Journal of Global Information Technology Management*, 150–168. doi:10.1080/1097198x.2014.951294

Foganholi, L. B., Garcia, E. R., Eler, D. M., Correia, R. C. M., & Junior, C. O. (2015). Supporting technical debt cataloging with TD-Tracker tool. *Advances in Software Engineering, 2015*, 1–12. doi:10.1155/2015/898514

Fusch, P. I., & Ness, L. R. (2015). Are we there yet? Data saturation in qualitative research. *Qualitative Report, 20*(9), 1408–1416. Retrieved from http://nsuworks.nova.edu/tqr/.

Gale, N. K., Heath, G., Cameron, E., Rashid, S., & Redwood, S. (2013). Using the framework method for the analysis of qualitative data in multi-disciplinary health research. *BMC Medical Research Methodology, 13*(1), 117. doi:10.1186/1471-2288-13-117

Giuffrida, R., & Dittrich, Y. (2013). Empirical studies on the use of social software in global software development–A systematic mapping study. *Information and Software Technology, 55*(7), 1143–1164. doi:10.1016/j.infsof.2013.01.004

Giuffrida, R., & Dittrich, Y. (2015). A conceptual framework to study the role of communication through social software for coordination in globally-distributed software teams. *Information and Software Technology, 63*, 11–30.

doi:10.1016/j.infsof.2015.02.013

Godin, J., & Goette, T. (2013). A pilot study of virtual teamwork training. *Communications of the IIMA, 13*(2), 29. Retrieved from http://scholarworks.lib.csusb.edu/ciima/.

Godin, J., Leader, L., Gibson, N., Marshall, B., Poddar, A., & Cardon, P. W. (2017). Virtual teamwork training: Factors influencing the acceptance of collaboration technology. *International Journal of Information and Communication Technology, 10*(1), 5–23. doi:10.1504/IJICT.2017.10001305

Greene, M. J. (2014). On the inside looking in: Methodological insights and challenges in conducting qualitative insider research. *Qualitative Report, 19*(29), 1–13. Retrieved from http://nsuworks.nova.edu/tqr/.

Guo, Y., Spinola, R. O., & Seaman, C. (2014). Exploring the costs of technical debt management - A case study. *Empirical Software Engineering, 21*(1), 159–182. doi:10.1007/s10664-014-9351-7

Gupta, S., Bhattacharya, V., & Singha, M. (2013). Pair programming "Potential benefits and threats". *International Journal of Advanced Computer Research, 3*(1), 108–113. Retrieved from http://accentsjournals.org/journals.php?journalsId=103.

Haahr, A., Norlyk, A., & Hall, E. O. (2014). Ethical challenges embedded in qualitative research interviews with close relatives. *Nursing Ethics, 21*(1), 6–15. doi:10.1177/0969733013486370

Hall, T., Min, Z., Bowes, D., & Yi, S. (2014). Some code smells have a significant but small effect on faults. *ACM Transactions on Software Engineering &*

Methodology, 23(4), 33:1–33:39. doi:10.1145/2629648

Hammer, M. J. (2016). Informed consent in the changing landscape of research. *Oncology Nursing Forum, 43*(5), 558. doi:10.1188/16.onf.558-560

Hanson, J. L., Balmer, D. F., & Giardino, A. P. (2011). Qualitative research methods for medical educators. *Academic Pediatrics, 11*(5), 375–386. doi:10.1016/j.acap.2011.05.001

Harvey, L. (2015). Beyond member-checking: A dialogic approach to the research interview. *International Journal of Research & Method in Education, 38*(1), 23–38. doi:10.1080/1743727X.2014.914487

Hayes, B., Bonner, A., & Douglas, C. (2013). An introduction to mixed methods research for nephrology nurses. *Renal Society of Australasia Journal, 9*(1), 8–14. Retrieved from https://www.renalsociety.org/.

Heikkila, V. T., Paasivaara, M., Lasssenius, C., Damian, D., & Engblom, C. (2017). Managing the requirements flow from strategy to release in large-scale agile development: A case study at Ericsson. *Empirical Software Engineering*, 1–45. doi:10.1007/s10664-016-9491-z

Hewitt, J. (2007). Ethical components of researcher-researched relationships in qualitative interviewing. *Qualitative Health Research, 17*(8), 1149–1159. doi:10.1177/1049732307308305

Hoare, Z., & Hoe, J. (2013). Understanding quantitative research: Part 2. *Nursing Standard, 27*(18), 48–55. doi:10.7748/ns2013.01.27.18.48.c9488

Holvitie, J., Leppanen, V., & Hyrynsalmi, S. (2014). Technical debt and the affect of

agile software development practices on it - An industry practitioner survey. In *Managing Technical Debt (MTD), 2014 Sixth International Workshop on* (pp. 35–42). IEEE. doi:10.1109/MTD.2014.8

Holvitie, J., & Leppanen, V. (2015). Examining technical debt accumulation in software implementations. *International Journal of Software Engineering and Its Applications, 9*(6), 109–124. doi:ijseia.2015.9.6.12

Hoyland, S., Hollund, J. G., & Olsen, O. E. (2015). Gaining access to a research site and participants in medical and nursing research: A synthesis of accounts. *Medical Education, 49*(2), 224–232. doi:10.1111/medu.12622

Huang, W.-H. D., Hood, D. W., & Yoo, S. J. (2013). Gender divide and acceptance of collaborative Web 2.0 applications for learning in higher education. *Internet and Higher Education, 16*, 57–65. doi:10.1016/j.iheduc.2012.02.001

Hussein, A. (2015). The use of triangulation in social sciences research: Can qualitative and quantitative methods be combined? *Journal of Comparative Social Work, 4*(1). Retrieved from http://journal.uia.no/index.php/JCSW.

Inayat, I., & Salim, S. S. (2015). A framework to study requirements-driven collaboration among agile teams: Findings from two case studies. *Computers in Human Behavior, 51, Part B*, 1367–1379. doi:10.1016/j.chb.2014.10.040

Islam, M. M. (2014). Dealing with Qualitative Methods in Bangladesh: The Potentials and Pitfalls of Research Design and Fieldwork. *Oriental Anthropologists, 14*(1), 13-26. Retrieved from http://www.printspublications.com/.

Izurieta, C., & Bieman, J. (2013). A multiple case study of design pattern decay, grime,

and rot in evolving software systems. *Software Quality Journal, 21*(2), 289–323. doi:10.1007/s11219-012-9175-x

Jacob, S. A., & Furgerson, S. P. (2012). Writing interview protocols and conducting interviews: Tips for students new to the field of qualitative research. *Qualitative Report, 17*(42), 1–10. Retrieved from http://nsuworks.nova.edu/tqr/.

Jayalakshmi, V. J., Kavitha, R. K., & Niroza, S. (2016). A study on pair programming effectiveness in a computer laboratory course. *International Journal of Science, Technology & Management, 5*(1). Retrieved from http://www.ijstm.com/.

Jordan, H. S. (2013). Maximizing sampling efficiency. *Applied Mathematics, 4*(11), 1547. doi:10.4236/am.2013.411209

Judkins-Cohn, T. M., Kielwasser-Withrow, K., Owen, M., & Ward, J. (2014). Ethical principles of informed consent: Exploring nurses' dual role of care provider and researcher. *Journal of Continuing Education in Nursing, 45*(1), 35–42. doi:10.3928/00220124-20131223-03

Jun, C.-J., Lee, J.-H., & Jeon, I.-S. (2014). Research about factor affecting the continuous use of cloud storage service: user factor, system factor, psychological switching cost factor. *Journal of Society for E-Business Studies, 19*(1). doi:10.7838/jsebs.2014.19.1.015

Kasemvilas, S., & Olfman, L. (2013). Improvement of MediaWiki to support mandatory collaboration. *Interactive Technology and Smart Education, 10*(3), 230–246. doi:10.1108/itse-05-2013-0009

Kavoura, A., & Bitsani, E. (2014). Methodological considerations for qualitative

communication research. *Procedia - Social and Behavioral Sciences, 147*, 544–549. doi:10.1016/j.sbspro.2014.07.156

Kemparaj, U., & Chavan, S. (2013). Qualitative research: A brief description. Indian *Journal of Medical Sciences, 67*(3-4), 89–98. doi:10.4103/0019-5359.121127

Khan, R. A., & Khan, S. U. (2017). Empirical exploration of communication and coordination practices in offshore software development outsourcing. *Proceedings of the Pakistan Academy of Sciences: A. Physical and Computational Sciences, 54*(1), 41.

Khayati, S., & Zouaou, S. K. (2013). Perceived usefulness and use of information technology: The moderating influences of the dependence of a subcontractor towards his contractor. *Journal of Knowledge Management, Economics and Information Technology, 3*(6). Retrieved from http://www.scientificpapers.org/.

Kirkwood, A., & Price, L. (2013).Examining some assumptions and limitations of research on the effects of emerging technologies for teaching and learning in higher education. *British Journal of Educational Technology, 44*, 536-543. doi:10.1111/bjet.12049

Kish, L. (1979). Samples and censuses. *International Statistical Review*, 99–109. Retrieved from http://www.isi-web.org/.

Kish, L., & Verma, V. (1986). Complete censuses and samples. *Journal of Official Statistics, 2*(4), 381–395. Retrieved from http://www.degruyter.com/view/j/jos.

Koelsch, L. E. (2013). Reconceptualizing the member check interview. *International Journal of Qualitative Methods, 12*(1), 168–179.

doi:10.1177/160940691301200105

Krishna, V., & Basu, A. (2015). A systematic method to evaluate the software engineering practices for minimizing technical debt. *International Journal of Computer Applications, 131*(4), 21–25. doi:10.5120/ijca2015907295

Lala, G. (2014). The emergence and development of the technology acceptance model (TAM). *Marketing From Information to Decision, (7)*, 149–160. Retrieved from http://econ.ubbcluj.ro/mid/.

Lamb, D. (2013a). Promoting the case for using a research journal to document and reflect on the research experience. *The Electronic Journal of Business Research Methods, 11*(2), 84–91. Retrieved from http://www.ejbrm.com/.

Lamb, D. (2013b). Research in the first person: Reflection on the research experience using a research journal. *Market & Social Research, 21*(2). Retrieved from https://www.amsrs.com.au/.

Largent, E., Grady, C., Miller, F. G., & Wertheimer, A. (2013). Misconceptions about coercion and undue influence: reflections on the views of IRB members. *Bioethics, 27*(9), 500–507. doi: 10.1111/j.1467-8519.2012.01972.x

Lavhengwa, T. J., van der Walt, J. S., & Lavhengwa, E. M. (2014). Factors influencing e-collaboration for knowledge development and innovation. *South African Journal of Information Management, 16*(1), 1–8. doi:10.4102/sajim.v16i1.588

Lee, D. Y., & Lehto, M. R. (2013). User acceptance of YouTube for procedural learning: An extension of the Technology Acceptance Model. *Computers & Education, 61*, 193–208. doi:10.1016/j.compedu.2012.10.001

Lesser, E., & Ban, L. (2016). How leading companies practice software development and delivery to achieve a competitive edge. *Strategy & Leadership, 44*(1), 41–47. doi:10.1108/SL-11-2015-0083

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software, 101*, 193–220. doi:10.1016/j.jss.2014.12.027

Li, Z., Liang, P., & Avgeriou, P. (2015). Architectural technical debt identification based on architecture decisions and change scenarios. In *Software Architecture (WICSA), 2015 12th Working IEEE/IFIP Conference on* (pp. 65–74). IEEE. doi:10.1109/WICSA.2015.19

Li, Z., Liang, P., Avgeriou, P., Guelfi, N., & Ampatzoglou, A. (2014). An empirical investigation of modularity metrics for indicating architectural technical debt. In *Proceedings of the 10th International ACM Sigsoft Conference on Quality of Software Architectures* (pp. 119–128). Marcq-en-Bareul, France: ACM. doi:10.1145/2602576.2602581

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic Inquiry*. Newbury, CA: Sage Publications.

Licorish, S. A., & MacDonell, S. G. (2014). Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: A longitudinal study. *Information and Software Technology, 56*(12), 1578–1596. doi:10.1016/j.infsof.2014.02.004

Linhares, G. B. R., Borges, M. R. S., & Antunes, P. (2012). Collaboration and conflict in

software review meetings. *International Journal of Information Technology & Decision Making, 11*(6), 1065–1085. doi:10.1142/s0219622012400159

Lopez-Martín, C., Nassif, A. B., & Abran, A. (2017). A training process for improving the quality of software projects developed by a practitioner. *Journal of Systems and Software, 131*, 98–111. doi:10.1016/j.jss.2017.05.050

Lucas, S. R. (2014). Beyond the existence proof: Ontological conditions, epistemological implications, and in-depth interview research. *Quality & Quantity, 48*(1), 387–408. doi:10.1037/a0038087

MacCormack, A., & Sturtevant, D. J. (2016). Technical debt and system architecture: The impact of coupling on defect-related activity. *Journal of Systems and Software.* doi:10.1016/j.jss.2016.06.007

Magdaleno, A. M., de Oliveira Barros, M., Werner, C. M. L., de Araujo, R. M., & Batista, C. F. A. (2015). Collaboration optimization in software process composition. *Journal of Systems and Software, 103*, 452–466. doi:10.1016/j.jss.2014.11.036

Malterud, K., Siersma, V. D., & Guassora, A. D. (2015). Sample size in qualitative interview studies: Guided by information power. *Qualitative Health Research*, 1–8. doi:10.1177/1049732315617444

Mangalaraj, G., Nerur, S., Mahapatra, R., & Price, K. H. (2014). Distributed cognition in software design: An experimental investigation of the role of design patterns and collaboration. *MIS Quarterly, 38*(1), 249–A5. Retrieved from http://www.misq.org/.

Marshall, B., Cardon, P., Poddar, A., & Fontenot, R. (2013). Does sample size matter in qualitative research?: A review of qualitative interviews in IS research. *Journal of Computer Information Systems, 54*(1), 11–22. doi:10.1080/08874417.2013.11645667

Martinez, Y., Cachero, C., & Melia, S. (2013). MDD vs. traditional software development: A practitioner's subjective perspective. *Information and Software Technology, 55*(2), 189–200. doi:10.1016/j.infsof.2012.07.004

Martini, A., Bosch, J., & Chaudron, M. (2015). Investigating architectural technical debt accumulation and refactoring over time: A multiple-case study. *Information and Software Technology, 67*, 237–253. doi:10.1016/j.infsof.2015.07.005

McDermid, F., Peters, K., Jackson, D., & Daly, J. (2014). Conducting qualitative research in the context of pre-existing peer and collegial relationships. *Nurse Researcher, 21*(5), 28. doi:10.7748/nr.21.5.28.e1232

McIntosh, M. J., & Morse, J. M. (2015). Situating and constructing diversity in semi-structured interviews. *Global Qualitative Nursing Research, 2*, 1–12. doi:10.1177/2333393615597674

McIntosh, S., Kamei, Y., Adams, B., & Hassan, A. E. (2014). The impact of code review coverage and code review participation on software quality: A case study of the QT, VTK, and ITK projects. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (pp. 192–201). Hyderabad, India: ACM. doi:10.1145/2597073.2597076

McIntosh, S., Kamei, Y., Adams, B., & Hassan, A. E. (2015). An empirical study of the

impact of modern code review practices on software quality. *Empirical Software Engineering*, 1–44. doi:10.1007/s10664-015-9381-9

Mealer, M., & Jones RN, J. (2014). Methodological and ethical issues related to qualitative telephone interviews on sensitive topics. *Nurse Researcher, 21*(4), 32. doi:10.7748/nr2014.03.21.4.32.e1229

Menolli, A., Cunha, M. A., Reinehr, S., & Malucelli, A. (2015). "Old" theories, "New" technologies: Understanding knowledge sharing and learning in Brazilian software development companies. *Information and Software Technology, 58*, 289–303. doi:10.1016/j.infsof.2014.07.008

Milovanovic, M., Minovic, M., Stavljanin, V., Savkovic, M., & Starcevic, D. (2012). Wiki as a corporate learning tool: Case study for software development company. *Behaviour & Information Technology, 31*(8), 767–777. Retrieved from http://www.tandfonline.com/loi/tbit20.

Molina, A. I., Gallardo, J., Redondo, M. A., & Bravo, C. (2015). Assessing the awareness mechanisms of a collaborative programming support system. *Dyna, 82*(193), 212–222. doi:10.15446/dyna.v82n193.53497

Morrison, P., Smith, B. H., & Williams, L. (2017). Surveying security practice adherence in software development. In *Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp* (pp. 85–94). Hanover, MD, USA: ACM. doi:10.1145/3055305.3055312

Motherudin, F., & Md. Moksen, N. E. (2015). A proposed model for code quality maturity model. *Journal of Software, 10*(3), 374–383. doi:10.17706/jsw.10.3.374-

383

Moylan, C. A., Derr, A. S., & Lindhorst, T. (2015). Increasingly mobile: How new technologies can enhance qualitative research. *Qualitative Social Work, 14*(1), 36–47. doi:10.1177/1473325013516988

Nel, G., Nel, L., & Cronje, J. (2016). Attributes contributing to students' use of quality software development practices. *African Journal of Information and Communication, 38*. doi:10.23962/10539/20329

Noble, H., & Smith, J. (2014). Qualitative data analysis: a practical example. *Evidence Based Nursing, 17*(1), 2–3. doi:10.1136/eb-2013-101603

Onwuegbuzie, A. J., & Byers, V. T. (2014). An exemplar for combining the collection, analysis, and interpretations of verbal and nonverbal data in qualitative research. *International Journal of Education, 6*(1), p183–p246. doi:10.5296/ije.v6i1.4399

Overhage, S., & Schlauderer, S. (2012). Investigating the long-term acceptance of agile methodologies: An empirical study of developer perceptions in scrum projects. In *System Science (HICSS), 2012 45th Hawaii International Conference on* (pp. 5452–5461). IEEE. doi:10.1109/HICSS.2012.387

Ozer, M., & Vogel, D. (2015). Contextualized relationship between knowledge sharing and performance in software development. *Journal of Management Information Systems, 32*(2), 134–161. doi:10.1080/07421222.2015.1063287

Park, N., Rhoads, M., Hou, J., & Lee, K. M. (2014). Understanding the acceptance of teleconferencing systems among employees: An extension of the technology acceptance model. *Computers in Human Behavior, 39*, 118–127.

doi:10.1016/j.chb.2014.05.048

Patton, M. Q. (1999). Enhancing the quality and credibility of qualitative analysis. *Health Services Research, 34*(5), 1189–1208. Retrieved from http://www.hsr.org/.

Peredaryenko, M. S., & Krauss, S. E. (2013). Calibrating the human instrument: Understanding the interviewing experience of novice qualitative researchers. *Qualitative Report, 18*(43), 1–17. Retrieved from http://nsuworks.nova.edu/tqr/.

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. doi:10.1177/1094428116629218

Plamondon, K. M., Bottorff, J. L., & Cole, D. C. (2015). Analyzing data generated through deliberative dialogue: Bringing knowledge translation into qualitative analysis. *Qualitative Health Research, 25*(11), 1529–1539. doi:10.1177/1049732315581603

Plonka, L., Sharp, H., van der Linden, J., & Dittrich, Y. (2015). Knowledge transfer in pair programming: An in-depth analysis. *International Journal of Human-Computer Studies, 73*, 66–78. doi:10.1016/j.ijhcs.2014.09.001

Polancic, G., Jost, G., & Hericko, M. (2015). An experimental investigation comparing individual and collaborative work productivity when using desktop and cloud modeling tools. *Empirical Software Engineering, 20*(1), 142–175. doi:10.1007/s10664-013-9280-x

Priyanka, S., & Kumar, A. (2013). Understanding the evolution of technology acceptance model. *International Journal of Advance Research in Computer Science and*

Management Studies, 1(6), 144–148. Retrieved from http://www.ijarcsms.com/.

Qiu, S., Wang, P., & Yang, P. (2015). The impact of personal psychology and behavior factors on the innovation assimilation of secure system development. *American Journal of Industrial and Business Management, 5*(04), 181. doi:10.4236/ajibm.2015.54020

Qu, S. Q., & Dumay, J. (2011). The qualitative research interview. *Qualitative Research in Accounting & Management, 8*(3), 238–264. doi:10.1108/11766091111162070

Ramasubbu, N., & Kemerer, C. F. (2014). Managing technical debt in enterprise software packages. *IEEE Transactions on Software Engineering, 40*(8), 758–772. doi:10.1109/TSE.2014.2327027

Ramasubbu, N., & Kemerer, C. F. (2015). Technical debt and the reliability of enterprise software systems: A competing risks analysis. *Management Science, 62*(5), 1487–1510. doi:10.1287/mnsc.2015.2196

Ramasubbu, N., Kemerer, C. F., & Hong, J. (2012). Structural complexity and programmer team strategy: An experimental test. *IEEE Transactions on Software Engineering, 38*(5), 1054–1068. doi:10.1109/TSE.2011.88

Ramasubbu, N., Kemerer, C. F., & Woodard, C. J. (2015). Managing technical debt: Insights from recent empirical evidence. *IEEE Software, 32*(2), 22–25. doi:10.1109/MS.2015.45

Rana, N. P., Dwivedi, Y. K., & Williams, M. D. (2013). E-government adoption research: An analysis of the employee's perspective. *International Journal of Business Information Systems, 14*(4), 414–428. doi:10.1504/ijbis.2013.057497

Reybold, L. E., Lammert, J. D., & Stribling, S. M. (2013). Participant selection as a conscious research method: Thinking forward and the deliberation of "Emergent" findings. *Qualitative Research, 13*(6), 699–716. doi:10.1177/1468794112465634

Riemenschneider, C. K., Hardgrave, B. C., & Davis, F. D. (2002). Explaining software developer acceptance of methodologies: A comparison of five theoretical models. *IEEE Transactions on Software Engineering, 28*(12), 1135–1145. doi:10.1109/TSE.2002.1158287

Rimando, M., Brace, A., Namageyo-Funa, A., Parr, T. L., Sealy, D.-A., Davis, T. L., … Christiana, R. W. (2015). Data collection challenges and recommendations for early career researchers. *Qualitative Report, 20*(12), 2025. Retrieved from http://nsuworks.nova.edu/tqr/.

Ritchie, J., Lewis, J., Nicholls, C. M., & Ormston, R. (2013). *Qualitative research practice: A guide for social science students and researchers* (2nd ed.). Sage.

Roberts, T. (2013). Understanding the research methodology of interpretative phenomenological analysis. *British Journal of Midwifery, 21*(3), 215–218 4p. doi:10.12968/bjom.2013.21.3.215

Robinson, O. C. (2014). Sampling in Interview-Based Qualitative Research: A Theoretical and Practical Guide. *Qualitative Research in Psychology, 11*(1), 25–41. doi:10.1080/14780887.2013.801543

Rodrigues, L. F., Oliveira, A., & Costa, C. J. (2016). Does ease-of-use contributes to the perception of enjoyment? A case of gamification in e-banking. *Computers in Human Behavior, 61*, 114–126. doi:10.1016/j.chb.2016.03.015

Rola, P., Kuchta, D., & Kopczyk, D. (2016). Conceptual model of working space for Agile (Scrum) project team. *Journal of Systems and Software, 118*, 49–63. doi:10.1016/j.jss.2016.04.071

Roulston, K., & Shelton, S. A. (2015). Reconceptualizing bias in teaching qualitative research methods. *Qualitative Inquiry, 21*(4), 332–342. doi:10.1177/1077800414563803

Sanjari, M., Bahramnezhad, F., Fomani, F. K., Shoghi, M., & Ali Cheraghi, M. (2014). Ethical challenges of researchers in qualitative studies: The necessity to develop a specific guideline. *Journal of Medical Ethics & History of Medicine, 7*(14), 1–6. Retrieved from http://jmehm.tums.ac.ir/.

Santos, V., Goldman, A., & De Souza, C. R. (2015). Fostering effective inter-team knowledge sharing in agile software development. *Empirical Software Engineering, 20*(4), 1006–1051. doi:10.1007/s10664-014-9307-y

Sedgwick, P. (2013). Convenience sampling. *BMJ, 347*. doi:10.1136/bmj.f6304

Shrivastava, S. V., & Rathod, U. (2017). A risk management framework for distributed agile projects. *Information and Software Technology, 85*, 1–15. doi:10.1016/j.infsof.2016.12.005

Stern, C., Jordan, Z., & McArthur, A. (2014). Developing the review question and inclusion criteria. *American Journal of Nursing, 114*(4), 53–56. doi:10.1097/01.NAJ.0000445689.67800.86

Stevens, M. R., Lyles, W., & Berke, P. R. (2014). Measuring and reporting intercoder reliability in plan quality evaluation research. *Journal of Planning Education and*

Research, 34(1), 77–93. doi:10.1177/0739456X13513614

Storey, M. A., Zagalsky, A., Filho, F. F., Singer, L., & German, D. M. (2017). How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering, 43*(2), 185–204. doi:10.1109/TSE.2016.2584053

Sundaramoorthy, V., & Bharathi, B. (2016). Need for social media approach in software development. *Indian Journal of Science and Technology, 9*(21). doi:10.17485/ijst/2016/v9i21/95271

Sutton, J., & Austin, Z. (2015). Qualitative research: Data collection, analysis, and management. *Canadian Journal of Hospital Pharmacy, 68*(3). doi:10.4212/cjhp.v68i3.1456

Svendsen, G. B., Johnsen, J.-A. K., Almas-Sorensen, L., & Vitterso, J. (2013). Personality and technology acceptance: The influence of personality factors on the core constructs of the technology acceptance model. *Behaviour & Information Technology, 32*(4), 323–334. doi:10.1080/0144929X.2011.553740

Tang, A., & Lau, M. F. (2014). Software architecture review by association. *Journal of Systems and Software, 88*, 87–101. doi:10.1016/j.jss.2013.09.044

Tang, F. (2015). When communication quality is trustworthy? Transactive memory systems and the mediating role of trust in software development teams. *R&D Management, 45*(1), 41–59. doi:10.1111/radm.12051

Thakurta, R., & Roy, R. (2012). Determinants of user involvement in software projects. In *System Science (HICSS), 2012 45th Hawaii International Conference on* (pp.

4894–4903). IEEE. doi:10.1109/HICSS.2012.203

Tom, E., Aurum, A., & Vidgen, R. (2013). An exploration of technical debt. *Journal of Systems and Software, 86*(6), 1498–1516. doi:10.1016/j.jss.2012.12.052

Tuohy, D., Cooney, A., Dowling, M., Murphy, K., & Sixsmith, J. (2013). An overview of interpretive phenomenology as a research methodology. *Nurse Researcher, 20*(6), 17. doi:10.7748/nr2013.07.20.6.17.e315

U.S. Department of Health & Human Services. (1979). *The Belmont Report*. Retrieved from http://www.hhs.gov/ohrp/humansubjects/guidance/belmont.html

Vahasantanen, K., & Saarinen, J. (2013). The power dance in the research interview: Manifesting power and powerlessness. *Qualitative Research, 13*(5), 493–510. doi:10.1177/1468794112451036

Vaismoradi, M., Jones, J., Turunen, H., & Snelgrove, S. (2016). Theme development in qualitative content analysis and thematic analysis. *Journal of Nursing Education and Practice, 6*(5), 100-110. doi:10.5430/jnep.v6n5p100

Venkatesh, V., & Davis, F. D. (2000). A theoretical extension of the technology acceptance model: Four longitudinal field studies. *Management Science, 46*(2), 186. doi:10.1287/mnsc.46.2.186.11926

Venkatesh, V., Brown, S. A., & Bala, H. (2013). Bridging the qualitative-quantitative divide: Guidelines for conducting mixed methods research in information systems. *MIS Quarterly, 37*(1), 21–54. Retrieved from http://www.misq.org/.

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. *MIS Quarterly 27(3)*, 425–478.

Retrieved from http://www.misq.org/.

Vicary, S., Young, A., & Hicks, S. (2016). A reflective journal as learning process and contribution to quality and validity in interpretative phenomenological analysis. *Qualitative Social Work*. doi:10.1177/1473325016635244

Vijayasarathy, L., & Turk, D. (2012). Drivers of agile software development use: Dialectic interplay between benefits and hindrances. *Information and Software Technology, 54*(2), 137–148. doi:10.1016/j.infsof.2011.08.003

Walker, J. L. (2012). The use of saturation in qualitative research. *Canadian Journal of Cardiovascular Nursing, 22*(2), 37–46. Retrieved from http://www.cccn.ca/.

Wallace, L. G., & Sheetz, S. D. (2014). The adoption of software measures: A technology acceptance model (TAM) perspective. *Information & Management, 51*(2), 249–259. doi:10.1016/j.im.2013.12.003

Wolgemuth, J. R. (2014). Analyzing for critical resistance in narrative research. *Qualitative Research, 14*(5), 586–602. doi:10.1177/1468794113501685

Wong, W. E., Li, X., Laplante, P. A., & Siok, M. (2017). Be more familiar with our enemies and pave the way forward: A review of the roles bugs played in software failures. Journal of Systems and Software. doi:10.1016/j.jss.2017.06.069

Wynn, D., & Williams, C. K. (2012). Principles for conducting critical realist case study research in information systems. *MIS Quarterly, 36*(3), 787–810. Retrieved from http://www.misq.org/.

Xiang, C., Lu, Y., & Gupta, S. (2013). Knowledge sharing in information system development teams: examining the impact of shared mental model from a social

capital theory perspective. *Behaviour & Information Technology, 32*(10), 1024–1040. doi:10.1080/0144929x.2012.745901

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

Yin, R. K. (2013). Validity and generalization in future case study evaluations. *Evaluation, 19*(3), 321–332. doi:10.1177/1356389013497081

Yli-Huumo, J., Maglyas, A., & Smolander, K. (2016). How do software development teams manage technical debt?–An empirical study. *Journal of Systems and Software*. doi:10.1016/j.jss.2016.05.018

Yucel, U. A., & Gulbahar, Y. (2013). Technology acceptance model: A review of the prior predictors. *Journal of Faculty of Educational Sciences, 46*(1), 89–109. doi:10.1501/egifak_0000001275

Zazworka, N., Vetro, A., Izurieta, C., Wong, S., Cai, Y., Seaman, C., & Shull, F. (2014). Comparing four approaches for technical debt identification. *Software Quality Journal, 22*(3), 403–426. doi:10.1007/s11219-013-9200-8

Zohrabi, M. (2013). Mixed method research: Instruments, validity, reliability and reporting findings. *Theory and Practice in Language Studies, 3*(2), 254–262. doi:10.4304/tpls.3.2.254-262

Appendix A: Human Subject Research Certificate of Completion

Certificate of Completion

The National Institutes of Health (NIH) Office of Extramural Research certifies that **Jeffrey Miko** successfully completed the NIH Web-based training course "Protecting Human Research Participants".

Date of completion: 01/18/2015.

Certification Number: 1653141.

Appendix B: Interview Protocol

Interview: Exploring Collaboration Strategies to Reduce Technical Debt

A. Introduce myself to the participant and thank them for participating.

B. Verified receipt of consent form, answer any questions and/or concerns of participant.

C. Remind participants I will be recording the interview and the interview will remain strictly confidential.

D. Turn on the recording device and announce the participant's identifying code, the date and time of the interview.

E. Start interview with the first question and continue through to the last question. Allow the participant to respond to each question and ask additional probing questions as necessary.

   1. What is your current position and role?
   2. How long have you been in your current position?
   3. How many years of experience do you have in software development?
   4. What degrees and industry certifications do you possess?
   5. How would you describe collaboration and its purpose in software development? What are the benefits of collaboration to your software development team and your organization?
   6. What are the methods and tools your team uses to facilitate collaboration? How would you describe the usefulness of those method and tools? How easy are those methods and tools to use?

7. What collaboration strategies does your team use to ensure programming logic meets requirements, software designs are accurate and programming code is free of defects? How would you describe the usefulness of those strategies to the overall success of projects? How easy are those strategies to implement?
8. What collaboration strategies does your team use to ensure team members follow your software development processes, policies and best practices? How would you describe the usefulness of those strategies in preventing future bug fixing, code refactoring and design changes? How easy are those strategies to implement?
9. How would you describe technical debt and its effects on software development projects? What are the largest sources of technical debt in your organization? How does your team manage technical debt?
10. How would you describe the collaboration strategies your team uses to identify, prevent and reduce technical debt? How would you describe the usefulness of those strategies in managing your technical debt? Which strategies are the most useful in minimizing technical debt? Which strategies are the easiest to implement?
11. What changes to your team's collaboration strategies do you feel would improve your team's ability to minimize or reduce technical debt in projects?

F. Ask the participant if they want to share any more information about the topics.

G. Ask the participant if they are aware of any documentation that might be relevant to the topics discussed.

H. Explain the concept of member checking and schedule a follow-up interview to review my interpretations with them.

I. Discontinue the audio record by turning off the device.

J. Thank the participant for partaking in the study. Confirm the participant has my contact information for any follow up questions and concerns.

Appendix C: Permission to Use Figures

Institute for Operations Research and the Management Sciences (INFORMS)
LICENSE
TERMS AND CONDITIONS

Apr 03, 2016

License Number	3841431077105
License date	Apr 03, 2016
Licensed content publisher	Institute for Operations Research and the Management Sciences (INFORMS)
Licensed content publication	Management Science
Licensed content title	User Acceptance of Computer Technology: A Comparison of Two Theoretical Models
Licensed copyright line	Copyright © 1989, INFORMS
Licensed content author	Fred D. Davis, Richard P. Bagozzi, Paul R. Warshaw
Licensed content date	08/01/1989
Volume number	35
Issue number	8
Type of Use	Thesis/Dissertation
Requestor type	Student
The intended publisher of new work	n/a
Format	Print, Electronic
Portion	chart/graph/table/figure
Number of charts/graphs/tables/figures	1
Rights for	Main product
Duration of use	None
Creation of copies for the disabled	no
With minor editing privileges	no
For distribution to	Worldwide
In the following language(s)	Original language of publication
With incidental promotional use	no
The lifetime unit quantity of new product	0 to 499
The requesting person/organization is:	Jeffrey Miko, DIT student at Walden University
Order reference number	None
Title of your thesis / dissertation	Exploring Collaboration Strategies to Reduce Technical Debt: A Case Study
Expected completion date	Mar 2017
Expected size (number of pages)	110
Total	0.00 USD

Institute for Operations Research and the Management Sciences (INFORMS) LICENSE TERMS AND CONDITIONS

Apr 01, 2016

License Number	3840230263331
License date	Apr 01, 2016
Licensed content publisher	Institute for Operations Research and the Management Sciences (INFORMS)
Licensed content publication	Management Science
Licensed content title	A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies
Licensed copyright line	Copyright © 2000, INFORMS
Licensed content author	Viswanath Venkatesh, Fred D. Davis
Licensed content date	02/01/2000
Volume number	46
Issue number	2
Type of Use	Thesis/Dissertation
Requestor type	Student
The intended publisher of new work	n/a
Format	Print, Electronic
Portion	chart/graph/table/figure
Number of charts/graphs/tables/figures	1
Rights for	Main product
Duration of use	None
Creation of copies for the disabled	no
With minor editing privileges	no
For distribution to	Worldwide
In the following language(s)	Original language of publication
With incidental promotional use	no
The lifetime unit quantity of new product	0 to 499
The requesting person/organization is:	Jeffrey Miko, DIT student at Walden University
Order reference number	None
Title of your thesis / dissertation	Exploring Collaboration Strategies to Reduce Technical Debt: A Case Study
Expected completion date	Mar 2017
Expected size (number of pages)	110
Total	0.00 USD

Appendix D: Interview Question Matrix

Interview questions	usefulness	ease of use	Intenion to use	Job relevance	Result demon-strability	Output quality	Volun-tariness	Social influe-nces	Usage behaviors
How would you describe collaboration and its purpose in software development?	X			X	X				X
What have been the benefits of collaboration to your software development team and your organization?	X			X	X	X			
What are the methods and tools your team uses to facilitate collaboration?	X		X	X	X				
How would you describe the usefulness of those method and tools?	X			X	X				
How easy have those methods and tools been to use?	X	X		X	X				
What collaboration strategies has your team used to ensure programming logic meets requirements, software designs are accurate and programming code is free of defects?	X	X		X	X	X		X	X
How would you describe the usefulness of those strategies to the overall success of your current projects?	X	X		X	X	X			
How easy were those strategies to implement?	X	X		X	X	X			
What collaboration strategies has your team used to ensure team members follow your software development processes, policies and best practices?			X				X	X	
How would you describe the usefulness of those strategies in preventing future bug fixing, code refactoring and design changes?			X				X	X	
How easy were those strategies to implement?		X	X				X	X	
How would you describe technical debt and its effects on software development projects?	X								X
What have been the largest sources of technical debt in your organization?	X			X					X
How has your team managed technical debt?	X					X			X
How would you describe the collaboration strategies your team has used to identify, prevent and reduce technical debt?	X		X						X
How would you describe the usefulness of those strategies in managing your technical debt?	X		X			X			X
Which strategies were the most useful in minimizing technical debt?	X		X		X				X
Which strategies were the easiest to implement?	X	X	X						X
Explain why you might have made changes to your team's collaboration strategies in the past	X		X						X
How these changes affected your team's ability to minimize or reduce technical debt in projects?	X		X		X				X