A COMMON STRATEGY FOR LEGACY SOFTWARE APPLICATIONS MODERNIZATION AMONG SMALL TO MID-SIZED SOFTWARE ENGINEERING ORGANIZATIONS: A QUALITATIVE DELPHI STUDY

by

Evelyn Esona Kiali Fomuso

A Dissertation Presented in Partial Fulfillment

Of the Requirements for the Degree

Doctor of Information Technology

Capella University

July 2019

Abstract

Changing business and technology needs pose a challenge for business-critical legacy software applications (LSAs) as most small to mid-sized software engineering organizations (SMSEOs) strive to maintain and grow their business. Such indispensable LSAs are hard to modernize due to the applications' complex nature, lack of documentation, incompatible, or un-migratable data. The present research highlights the knowledge gap that many information technology (IT) leaders in SMSEOs have no common strategy for modernizing their indispensable LSAs. The present study was conducted to identifying a common LSA modernization strategy, guided by the concepts of complexity, change management, and disruptive innovation theories. The overarching research question is: What common modernization strategy can IT leaders in SMSEOs leverage to modernize their indispensable LSAs as technology changes? The methodology used was qualitative with a three-round Delphi technique to gather and analyze the modernization experiences of IT leaders in SMSEOs. The sample size went from 13 to 10 SMSEO IT leaders in the three Delphi rounds. Round 1 data analysis using a combination of constant comparison analysis (CCompA), and classical content analysis (CContA) revealed six unique strategies from 13 responses, which converged into three in Round 2 with the majority preferred choice at 80%, and two in Round 3 at 90% for being LSA replacement with new development. The observed common strategy had pros such as comprehensive, flexible, customizable, adaptable, compatible, added value to the organization, organization autonomy, cost-effective because it can be paced and budgeted, fresh perspective, risk management, and traceability back to the organizations' business requirements. Comprehensive was the most sought-after aspect of a strategy. The participants agreed unanimously on the observations made from the data, and 90% agreed on a common strategy.

Dedication

I dedicate this study to the loving memory of my amazing daddy, late Mr. Titus N. Fomuso, who passed away shortly after I started this journey, for the life-long teaching and ethics that he instilled in me so that I could continue the journey even while in mourning. My dad battled with cancer for over 11years, during which no one could tell he was sick except for his close family, due to his joie de vivre. I also dedicate this work to my super mom, Mrs. Monica V. Fomuso (Mère), who has shown me how to be resilient during tough times, because those never last but tough people do. To my darling siblings (aka the TNMV-Fomuso-lings/Valentinelings): Ni Gang Fomuso (Ba Nkom Titus), Ma Kah Eniola, Ma Andin Fomuso, Ma Nandet Fomuso, Ma Ango Fomuso-Ekellem, and Dr. Lusia Fomuso (PharmD), my beloved nieces and nephews, and other close family and friends who encouraged and cheered me on during this journey. I love you all!

Acknowledgments

I thank God for the strength and good health during this mammoth journey. My heartfelt gratitude to my dissertation mentor and chair Dr. Vu Tran for your time, advice, encouragements, guidance, prompt responses and feedback during my whole DIT journey, may God continue to bless you tremendously. I much appreciate my committee members Dr. Isaac Gbenle and Dr. Oludotun Oni, for your constructive feedback that pushed me toward the right direction. I would like to acknowledge my employer PGi and my managers for all the support. Immense thanks to my whole cohort (The Opposite of Theory), for encouraging one another as we went through this process, especially my DIT accountability partner Ms. Angelia R. Childs. Special thanks to all the participants in my data collection process, you took time out of your busy schedule to respond to my multi-round essay questions, God bless you richly. Immense thanks to Dr. Lilian Fokwang for the impromptu professional editing of my dissertation. Dr. Comfort Akuh for being a sounding board when I was drowning in the stress of my school work and giving me sound advice that helped me get to the end, I say kuzong. To my big bro/grandfather Dr. William Dzekashu, who was the first Ph.D. I called when I decided to start this journey, for the sound advice that kept me going, njika njamu Ba. To my dear friend Dr. Tambei Chiawah (aka Boeuf), who was always ready to encourage and assist me during this journey, I say thanks so much. Dr. Lum Fube and Dr. Naomi Azunie, for being my standing powerhouse of advice as we went through the journey together, words can't express how thankful I am. Thanks to all my loved ones who have gone through a similar journey in terminal degrees, for helping to prepare me for the task at hand. Finally, to my parents, siblings, Mrs. Nange K-Mutia, and all my close family and friends, I say special thanks for all your prayers, moral support, and for cheering me on. God bless you all. Njika! Kuzong! Danke! Merci! Thank you!

Table of Contents

List of Tables

List of Figures

CHAPTER 1. INTRODUCTION

Introduction

Software applications that are current and relevant today can quickly become tomorrow’s legacy due to rapid changes in technology (Tantry, Murulidhar, & Chandrasekaran, 2017). While there are many definitions, for this qualitative Delphi study, a legacy software application (LSA) is a large and complex program, custom created based on outdated technologies, which lacks adequate documentation and resists modification, consequently it is inflexible to meet changing organizational needs (Crotty & Horrocks, 2017; Srinivas, Ramakrishna, Rao, & Suresh Babu, 2016; Stamford, 2014). As a result, organizations who operate LSAs that provide functionality for daily business-critical operations typically face steadily increasing challenges as time passes, and those LSAs become increasingly obsolete (Srinivas et al., 2016). Such challenges are usually directly related to organizations’ needs to address continually changing market conditions or business priorities (Islam, Toma, Selim, Gias, & Khaled, 2016; Jain & Chana, 2015).

Having a common LSA modernization strategy is a necessity for effective information technology (IT) management in organizations that heavily depend on software applications for business-critical operations (Srinivas et al., 2016). In the present study, LSA modernization is a set of both managerial and technical activities for replacing or transforming software, by applying modern technology, cost-effectively, without impacting the business service, to render the software not only reusable but also more valuable, profitable, reliable, and extend the functionality of the existing LSA’s output (Norfolk, 2014; Zheng, 2013). The aim is to enhance the application’s functionality, interoperability, performance, or reusability without impacting the business service that the application supports (Norfolk, 2014). Software application modernization spans both software maintenance, which involves the modification done after delivery to either correct, adapt, improve, or detect and correct potential faults (Zheng, 2013), and software migration which involves moving from one operating environment to a better one (Srinivas et al., 2016).

Chapter 1 is organized to introduce the topic of LSA modernization. After a review of the background and challenges of LSA modernization, Chapter 1 contains information on the research purpose, the research questions that inform the present study, and the study's rationale. Next, Chapter 1 outlines the theoretical frameworks that guided the research and the significance of the study. Finally, Chapter 1 outlines some essential terminologies used in the research, the study's assumptions and limitations, and the organization of the remainder of the study.

Background

LSAs play a vital role in the operations of many major organizations, and problems arise when organizations are unable to incorporate their changing business priorities smoothly into the existing software applications (Crotty & Horrocks, 2017). Often, the challenge is to maintain an indispensable LSA while making it more efficient by applying current technology, code, and programming languages (Beijert, 2016). LSAs are associated with a number of potential issues which include high support and maintenance costs, the potential for deficiencies in proper documentation for what are often highly complex programs, the potential for slow and inefficient run times, and challenges with updating the business logic by extending functionality to keep up with changing organizational and business-critical needs (Beijert, 2016; Crotty & Horrocks, 2017; Dedeke, 2012; Kaur, Ahamad, & Verma, 2016; Norfolk, 2014).

Guided by theories of disruptive innovation, change management, and complexity, this qualitative Delphi study aims to find common ground with regards to the modernization of LSAs by identifying commonalities in the modernization practices based on the experience of IT leaders and legacy application managers in small to mid-sized software engineering organizations (SMSEOs). Disruptive innovation is innovation that not only improves a market but can overshoot the needs of consumers while responding to disruptive threats and cause a disruption in that market by displacing established competitors in the market with a less expensive and accessible version of a product (King & Baatartogtokh, 2015).

Such disruptions introduced by new competitors could happen as a result of new technology which is periodic and threatens to destroy organizations that rely heavily on LSAs and cannot easily adopt the new technology (Bakhit, 2016; Christensen, 1997). Disruptive innovation research has shown that IT leaders sometimes fail to modernize LSAs, especially if customers have not demanded new technology (Sandström, Berglund, & Magnusson, 2014). When IT decision-makers decide to replace the LSA, valuable company data must be preserved to avoid adding the loss of data to the replacement cost (Crotty & Horrocks, 2017).

Ali and Lai (2015), noted that requirement change is inevitable within organizations, and one of the causes can be changes in technology. As organizations go through requirement changes, its culture, processes, and technologies need to change to adapt to emerging market needs, to continue functioning effectively (Lawrence, 2015). Modernizing existing LSAs is a challenge that needs careful planning. Organizations must evaluate all deciding factors in software modernization before adopting new technology (Oliveira, Thomas, & Espadanal, 2014). However, research informed by the complexity and change management theories, show that traditional change models are unreliable to IT leaders as they face the challenge of LSAs modernization, because of the impossibility to predict what happens in each phase of the change (Lawrence, 2015). As organizations grow and evolve, their culture, processes, and technologies become more complex as they transform to meet the changing market needs. While an organization's capacity to handle these changing needs, depends on a combination of organizational size, culture, processes, and technologies (Aguirre & Alpern, 2014), building organizational capacity to address changing needs remain significant to preparing for the future.

Measuring the complexity of LSAs is essential because of the need to predict what happens in each phase of the modernization (Lawrence, 2015). For instance, despite the availability of standardized software complexity measurement technologies such as McCabe's cyclomatic complexity, according to Kaur et al. (2016), there is a lack of consensus on how to apply software complexity measurement in software modernization strategies. Another influencing factor is the lack of understanding of the human perspective on the evolution of, as well as obstacles in, reusing software systems (Vogel-Heuser, Fay, Schaefer, & Tichy, 2015). Also, there is the aspect of determining if they trust the new technology to migrate to, as well as the assessment of the need, process, and cost of modernization (Crotty & Horrocks, 2017). Finally, the lack of a standardized strategy to enable efficient and effective modernization of software application is yet another critical challenge (Vogel-Heuser et al., 2015).

Business Technical Problem

LSAs are large and complex applications that are critical for business but resist modification (Crotty & Horrocks, 2017; Srinivas et al., 2016). The LSAs are typically created using outdated technologies, although the applications remain indispensable to the organizations because of the daily business-critical use (Rai, Sahoo, & Mehfuz, 2015; Srinivas et al., 2016). The failure of an indispensable LSA can have a significant impact on business (Crotty & Horrocks, 2017; Srinivas et al., 2016). Typically, with such indispensable LSAs, there is little or no documentation, and even when some documentation exists, the documentation does not provide reliable information, and so it is hard to understand or update the systems (Srinivas et al., 2016). LSAs pose some challenges to organizations such as slow speed, high maintenance costs and even costly fault detection due to obsolete technology; and for the organizations to modernize or refactor said applications, they need a Business Process Reengineering (BPR) (Srinivas et al., 2016). A BPR is a way to deal with change management where the related undertakings required to get a particular business result are fundamentally updated (Business process reengineering (BPR), n.d.).

The general IT problem contemplated in the present study is the lack of common strategies for modernizing indispensable LSAs as technology changes (Crotty & Horrocks, 2017). The specific IT problem is that many IT managers in SMSEOs have no common strategy for modernizing their indispensable business-critical LSAs as technology changes at a fast pace, to match the technological changes and handle their business challenges, causing the modernization process to be inefficient and costly (Crotty & Horrocks, 2017; Morton, Beckford, & Cooke, 2015; Rai et al., 2015).

Some examples of business-critical applications that are indispensable include the software used by banks to maintain customers finances, by hospitals to manage patient information, by schools for course details, student or prospective student information, or by software engineering organizations to fulfill daily business needs (Beijert, 2016). Due to rapid technology changes, IT managers need a strategy for modernizing their indispensable LSAs accordingly to match the technological changes and handle their business challenges (Dedeke, 2012; Islam et al., 2016; Jain & Chana, 2015; Morton et al., 2015; Serrano, Hernantes, & Gallardo, 2014). It is also vital that IT managers adopt a common practical modernization strategy that is secure, scalable and sustainable to support business-critical systems, and control the high costs associated with modernizing large-scale LSAs (Rai et al., 2015). Adopting a common modernization strategy would help the indispensable LSAs to maintain their relevance and usability in the organization. (Jain & Chana, 2015).

Research Purpose

The purpose of this qualitative Delphi study is to identify a common strategy for modernizing indispensable LSAs and create a modernization model (El-Gazzar, Hustad, & Olsen, 2016). Many IT managers in software engineering organizations have no common strategy for modernizing their indispensable business-critical LSAs as technology changes at a fast pace, causing the modernization process to be inefficient and costly (Crotty & Horrocks, 2017; Morton et al., 2015; Rai et al., 2015). Identifying a common modernization strategy would help the software engineering organizations to be able to plan for and budget for the modernization of their LSAs to boost revenue generation from the LSAs while continuing to add value to their business and to customer experience (Letier, Stefan, & Barr, 2014). Using the Delphi technique in this qualitative study may add to the scholarly knowledge base by providing perspectives from subject matter experts experienced in modernizing various types of LSAs (Maxey & Kezar, 2016). The outcome of this qualitative Delphi study enhances both the scholarly and practitioner knowledge-base on how to approach the modernization of LSAs, according to the perspectives of IT leaders (Beijert, 2016; Skulmoski, Hartman, & Krahn, 2007).

Research Question

The research question relates to the present business technical problem by analyzing the perspectives of IT leaders. IT leaders most affected by this qualitative Delphi study are those, in SMSEOs, who have no common strategy for modernizing their indispensable business-critical LSAs (Crotty & Horrocks, 2017; Morton et al., 2015; Rai et al., 2015). The overarching research question is: What common modernization strategy can IT leaders in SMSEOs leverage to modernize their indispensable LSAs as technology changes?

Rationale

From my professional experience with LSAs across several industries, every modernization project used a different strategy and approach, and each strategy had a number of pros and cons. After reading a number of professional and academic literature, it became clear to me that there was a gap in the knowledge base regarding a common modernization strategy and made this topic a great choice for my research study. This research also helps regularize the general performance expectation from the LSAs, as well as budgeting, by helping with the existing problems with LSAs such as lack of flexibility, lack of information, lack of resources, and high support and maintenance costs (Stamford, 2014). Regarding the cost impact of software there is always uncertainty, and therefore difficult to precisely quote how significant the effect of this problem is, but with LSA the monetary impact would be at least the income it already contributes to the organization, and even more impact for indispensable applications used for daily business-critical processes (Letier et al., 2014). Per Crotty and Horrocks (2017), organizations still relying on indispensable LSAs for their vital business, spend about 75% of their IT budget on maintaining such applications to keep them operational. According to OnBase (n.d.), about 70% of business systems in the corporate world are LSAs. Additionally, 70% of organizational software expenditure is spent on the maintenance of LSAs (Kaur et al., 2016). Also, per Marfatia (2014), 70% of business and government transactions are still processed in COBOL even though it originated in 1959.

The three scenarios above throw out the same number as the percentage, indicating that depending on how an LSA is defined almost every software engineering organization has LSAs which are critical for the business (Tantry et al., 2017). LSAs tend to have less secure technology or language, and there is a chance of exploiting the security loopholes (Sadeghi, Davi, & Larsen, 2015). Because the existing systems provide core functionalities to organizations, the impact of failure could mean the loss of business, individual or business data (Srinivas et al., 2016). To stay relevant in the market, firm IT decision-makers need to keep their LSAs flexible, adaptable, and current with the constant and fast change in technology accordingly (Serrano et al., 2014).

Theoretical Framework

The theories that informed this qualitative Delphi study include change management theory, disruptive innovation theory, and complexity theory. Kaur et al. (2016) discussed the measuring of the complexity of LSAs, which suggests that LSAs are in fact, complex applications. Given the fast pace of technology change, with the modernization of a legacy application, the change involved which would warrant a change management concept. Considering that highly innovative change in technology typically involves cost reduction, and also, since one major pushing point for considering the adoption of new technology in LSAs is cost, the disruptive innovation theory becomes a vital backing for the present research (Bakhit, 2016; Crotty & Horrocks, 2017). Complexity theory informs the present study because, complexity theory states that a system constitutes ongoing instability and entropy conditions and as a result, varying structures and patterns emerge as the system modernizes, which also ties in with change management and disruptive innovation. (Lowell, 2016; Zanotta, 2013).

Change Management Theory

All organizations go through changes at some point to stay competitive. The need to adapt to external changes, as well as the requirements for more effective organizational integration, drives evolution. When an organization changes, all critical aspects of the organization might be affected, including its missions, structures, IT infrastructures, and culture. Organizational changes might not succeed if critical components failed to change (Aguirre & Alpern, 2014). Organizational change can come in an indirect and imposed format due to circumstances, or it can be planned and then adopted to encourage improvement, growth, and acceptance which is where the role of change management comes in (Aser, 2015). Some examples of the change management model, process and plans include the Deming Cycle, John Kotter’s change management model, McKinsey’s change management framework, and the Prosci ADKAR process (Aser, 2015). Strategies for dealing with LSA modernization need to be part of an organization’s change management program.

Disruptive Innovation Theory

Disruptive innovation theory informs the present research because, organizations relying on indispensable LSAs for most of their critical daily business run the risk of being disrupted by new entrants in their niche market, if they cannot match the pace of technology change by effectively evolving their systems to meet new needs (Rai et al., 2015; Sandström et al., 2014). Most organizations tend to focus on the daily operations and easily miss the technological changes, both expected and unexpected, that warrant the modernization of their indispensable LSAs (Bharathy & McShane, 2014, p. 41). Also, the high maintenance cost of LSAs is a deciding factor in finding a common strategy for modernization. Therefore, organizations would need to factor in strategies for handling their LSAs to avoid the impact of disruptive innovation on the organization ("Disruptive innovations theory," n.d.). Some organizations prefer to leave their LSAs as they are, most possibly because they are satisfied with their clientele and do not want to incur unnecessary risk (Sandström et al., 2014). Meanwhile, others want to match technology changes and be as competitive as possible while opening their business to clients who do not have another option, and this qualitative Delphi study is basing on the latter set of organizations. Some case studies have shown that some large companies failed because of lack of a strategy to deal with smaller competitors with disruptive innovation and that the organizations must establish a change management strategy, to address strategies for effective and innovative modernization of the LSAs (Bakhit, 2016; Vecchiato, 2017).

Complexity Theory

George A. Cowan first founded complexity theory at the Santa Fe Institute in New Mexico, back in the 1980s, and has since been used as the framework of various qualitative studies (Dutcher, 2011; Giles, 2015). Giles (2015) also posits that complexity is an integral and unavoidable part of any IT project and considering that the act of modernizing LSAs software is an IT project, complexity theory is inescapable, and must the strategy factor it. Per Cooke-Davies (2011), the definition of complexity is very subjective and covers aspects such as supply chain, geography, time pressure, size and scale, stakeholders, and technology. According to Sargut (2011), the degree of complexity of a complex system may lie beyond cognitive limits, and even past behaviors may not help in determining future ones. Also, according to Cooke-Davies (2011), when a system consists of parts that are interdependent but not necessarily predictable, it is known as a complex system.

Srinivas et al. (2016) state that LSAs are large and complex existing application software in organizations that typically function as the backbone for the day-to-day running of the business. Unlike machines that are a sum of their parts and their functionality can be summed up as cause and effect, complex systems and non-reductive, non-linear, and have disproportionate cause-effect relationships, with new sets of models and methods of approach (Labs, 2014, 2017). Due to the complex nature of LSAs, it is hard to make decisions about migrating or maintaining them without, first of all, measuring the complexity and determining the best change management approach for the applications. Complexity theory is one of the bases of the present research because the definition of LSAs imply that such systems are complex (Srinivas et al., 2016). Complexity theory must be factored in, to determine if any strategies can be applied to making decisions related to modernizing a legacy application (Kaur et al., 2016). Complexity theory ties in with change management theory in the subject of modernizing indispensable legacy application because modernization is a change that needs management and LSAs are complex systems (Kaur et al., 2016; Lowell, 2016). Also, some studies present the application of complexity theory to change management in organizations (Lowell, 2016). Furthermore, in a complex organization with controlled behavior, and various independent actors are interacting with each other, leaders can apply complexity theory to encourage non-linear thinking and understand organizational change (Dutcher, 2011). Then, from the angle of disruptive innovation, some studies present why IT leaders fail to modernize LSAs, especially if customers have not demanded new technology (Sandström et al., 2014).

The three theories were used together as to inform the present qualitative study because disruptive innovation occurs because there is a change of some sort in an industry (Rai et al., 2015; Sandström et al., 2014). Then because LSAs are complex and their modernization involves a change of some sort as well, and the change could lead to disruptive innovation in SMSEOs, meaning there would be a need for change management (Rai et al., 2015; Sandström et al., 2014). Therefore, the three theories were deemed necessary in the processing of the business technical problem in the present study.

Significance

According to Beijert (2016), many organizational leaders are reluctant to take any action about problems with legacy application resulting in application software coming to the end-of-life status. The present study investigates the possibility of finding a common standardize-able modernization strategy for indispensable LSAs. Specifically, this qualitative Delphi study examines the perspectives and potential challenges of IT leaders in SMSEOs, regarding LSA modernization (Hakemi, Jeong, Ghani, & Sanaei, 2014; Serrano et al., 2014). Additionally, the present study seeks to identify a common strategy for modernizing such applications based on commonalities of experiences (Hakemi et al., 2014; Serrano et al., 2014).

Having a common modernization strategy in place is beneficial to the decision-makers of SMSEOs because they would be able to continually provide services to both internal clients and paying customers while also leveraging modern technology as it changes (Bakhit, 2016; Vecchiato, 2017). Also, providing a structure to use for training and knowledge transfer to new hires as needed (Dedeke, 2012). The present research may fill the knowledge gap by exposing the limitations involved in determining a common strategy. The study results would also provide a guide for IT managers in organizations relying on indispensable LSAs to determine when to cut their losses and completely replace application software as they become legacy and cost of maintenance becomes higher than the replacement cost. This qualitative Delphi study also helps regularize the general performance expectation from the LSAs, as well as budgeting. This research also may contribute to IT literature and knowledge on the challenges of software modernization (Jain & Chana, 2015). The findings from this qualitative Delphi study may add to the existing body of research knowledge regarding software modernization either by identifying a common LSA modernization strategy.

Definition of Terms

Complex systems. According to Cooke-Davies (2011), when a system consists of parts that are interdependent but not necessarily predictable, it is known as a complex system.

Complexity. The state or condition of having numerous parts that are hard to comprehend or find answers to ("Complexity," n.d.).

Complexity theory. The study of how designs, request, structure, and new themes emerge from clearly chaotic or complex systems and then again, how complex conduct and structure arise from basic and straightforward underlying rules (Cicmil, Cooke-Davies, Crawford, & Richardson, 2009).

Delphi technique. An exploratory qualitative technique, used for allowing a group of experts to deal with a problem in a structured communication setting such as a conference workshop (Deschene, 2016).

Disruptive innovation. An innovation that not only improves a market but can overshoot the needs of consumers while responding to disruptive threats and cause a disruption in that market by displacing establishing competitor in the market with a less expensive and accessible version of a product ("Disruptive innovations theory," n.d.; King & Baatartogtokh, 2015).

Indispensable LSAs. For this qualitative Delphi study, an indispensable legacy application is an application that an organization critically depends on daily and cannot quickly get rid of the application without the risk of a substantial loss or even closure of the business (Rai et al., 2015).

Innovation strategy. A proposition or a template of changes or novelty that would bring new value to business processes, services, or models and make a company more competitive ("Strategy innovation," n.d.).

Legacy software application (LSA). A large and complex program, custom created based on outdated technologies, which lacks adequate documentation and resists modification, consequently, it is inflexible to meet changing organizational needs (Crotty & Horrocks, 2017; Srinivas et al., 2016; Stamford, 2014).

Legacy software modernization. A set of both managerial and technical activities for replacing or transforming software, by applying modern technology, cost-effectively, not impacting the business service, to render the software not only reusable but also more valuable, profitable, and reliable (Norfolk, 2014; Zheng, 2013).

Software maintenance. Software application modification done after delivery to either correct, adapt, improve, or detect and correct potential faults (Zheng, 2013).

Software migration. According to ("Migration" n.d.), software migration is the act of moving from one operating environment to another one, which is mostly better than the previous one, as a way of modernization which could involve hardware or software, but this qualitative Delphi study focuses only on the software portion of this definition.

Assumptions and Limitations

Assumptions

This researcher assumed that the Delphi technique is the best for the present research because it is the technique for organizing an amass correspondence procedure to encourage aggregate critical thinking and to structure models (Linstone & Turoff, 2011). Another assumption was that the participants' experience with modernizing one or more LSAs is an adequate representation of SMSEOs, to provide suggestions for a common modernization strategy regardless of their job title. Another assumption was that the system modernization would include all that the system needs to function. If there are any data, hardware, or tools that the applications need to function, those would be considered, but the focus would be solely on the applications themselves. One additional addition assumption was that the participants were honest and truthful in their participation and in describing their experience.

Limitations

According to Kirkwood and Price (2013), researchers should acknowledge their underlying limitations as well as assumptions of their study to be able to interpret their findings appropriately. This research was able to identify the limitations that shape this qualitative Delphi study to include the scope of this present study which was limited to the process of modernizing the applications themselves and does not extend to the migration of data. Also, as a qualitative study based on purposive sampling, the findings of the study may not be generalizable to a broader population of organizations that need to modernize their legacy systems, and because there is no recommended sample for a homogeneous group a sample size of 10 to 15 might suffice (Skulmoski et al., 2007). Another limitation is that while many industries such as education, health, engineering, military, banking have LSAs, due to time and scope constraints, not all of the industries could be studied in the present research. Therefore, the present researcher sampled those IT leaders, including team leads, IT managers, IT directors, CTOs, CIOs, and other decision-makers of SMSEOs, who have modernized or were currently modernizing their indispensable LSAs.

Organization for Remainder of Study

Chapter 1 provided the introduction of the business IT problem, the background, rationale, significance, assumptions, and limitations as well as questions that guide the research. Chapter 2 contains the literature review of all articles related to either the business IT problem, theoretical framework, or the methodology used, grouped under subheadings indicating how they are related to the present dissertation and what area is supported. Chapter 3 outlines the research methodology, design, technique, population, sample size, and the credibility and dependability of the method. Also, Chapter 3 addresses the setting, data collection, along with data analysis and ethical considerations, as well as an analysis of the research question, sub-questions. Chapter 4 presents the data collection process, results of the data collection, analysis of the results for evolving concepts, and a summary of the findings. Chapter 5 covers the evaluation of the research questions and the research purpose fulfillment recommendations for future studies and overall conclusions, as well as the contribution of the present research to the business IT problem, lessons learned, and recommendations for further studies.

CHAPTER 2. LITERATURE REVIEW

Introduction

Chapter 2 contains the synthesis of the literature on the legacy application modernization problem, the theoretical framework that drives the study as well as discussion on how IT managers have been approaching LSA modernization issue. Chapter 2 also includes literature on the theories backing the present qualitative study, which include change management theory and how the theory applies to legacy application modernization, complexity theory, as well as the disruptive innovation theory. Also, it contains literature that supports the research methodology and technique for this qualitative Delphi study. Existing literature presents the term legacy software applications (LSAs) interchangeably with legacy applications, legacy software, legacy code, legacy application software, legacy application systems, and legacy software system. However, for consistency sake, only the term legacy software applications (LSAs) is used in the present qualitative study.

Techniques of Literature Search

I reviewed the literature for this qualitative Delphi study, ensuring that the articles included are predominantly scholarly and not older than five years. Data sources for locating the articles include the Capella Summons (Capella University Library), Google, the general Google Scholar, and the dedicated Capella Google Scholar. I also used ProQuest Central, and eBook Collection (EBSCOhost) to search for and collect references. I followed cited references, mined the bibliographies, as well as profile-searched the articles and authors found to uncover more references.

Additionally, the databases that the present researcher used include ACM Digital Library, Computers & Applied Sciences Complete, CQ Researcher, Dissertations @ Capella, Dissertations and Theses Global, EBSCOhost, and PsycTESTS. This researcher used the advanced search function where ever the present researcher could to limit the articles to those from 2014 and newer. The search keywords included legacy application, defining legacy applications, replacing legacy systems, reliability, migration cost, maintenance cost, legacy systems, application software, legacy application software, legacy code, migration of LSAs, change management, change management theory, complexity theory, disruptive innovation theory, Delphi methodology, Delphi technique, Delphi, qualitative research, software modernization, modernization strategies, application modernization, and Delphi dissertations Capella. Whenever the search terms brought about an excessive number of articles, the present researcher restricted the results by time restriction, meaning articles from the last three years on a particular topic.

Theoretical Framework

The three theories that frame this qualitative Delphi study are: change management, disruptive innovation, and complexity theories. Complexity is one of the essential aspects of an IT project, and the organizational understanding, framework, and environment must change for project teams to strategize and simplify project management processes adequately. The need to adapt to external changes as well as the need for more effective organizational integration drive these changes.

When an organization changes, all critical aspects of the organization might be affected, including its missions, structures, IT infrastructures, and culture. Organizational changes might not succeed if critical components failed to change (Giles, 2015). Changes that an organization makes to application software for modernization purposes are usually innovative and could quickly lead to disruptive innovation. An LSA modernization strategy needs to be part of an organization's change management program. Since the purpose of this qualitative Delphi study is to explore the possibility of finding a common modernization strategy for LSAs, all three of these theories make for an excellent framework for this Delphi research. This literature review begins with a more in-depth examination of the three theories that guide the present research.

Change Management Theory

The change management theory is sometimes known as the change theory and was founded by Kurt Lewin (Cummings, Bridgman, & Brown, 2016; Hartzell, 2017; Petiprin, 2016). According to Cummings et al. (2016), some scholars argue that Lewin oversimplified the change process, although there is also ample literature that supports the significance of Lewin's model. Lewin's model is also otherwise called changing as three steps. The three steps are unfreezing, changing, and refreezing, derived from the process followed when preparing a meal from frozen supplies (Cummings et al., 2016; Hartzell, 2017; Petiprin, 2016).

Unfreezing. This phase is the initial stage of making any change in an organization. In this stage, the problem that the change is addressing is presented to make everyone aware of the situation, and also represents getting all stakeholders comfortable with the fact that change is needed since people tend to resist change (Hartzell, 2017). In this phase, communication is crucial to get all concerned parties convinced of the necessity of the change to the point when they realize the importance and urgency of the imminent change. When considering LSAs, this is the stage in which the organization would have to come to terms with the fact that their application is indeed legacy and warrants looking into (Beijert, 2016).

The organization would also have to determine how changing technology has affected the system before they determine how it should be modernized (Luftman & Kempaiah, 2007). Figuring out how to execute this phase is vital to the success of the change process. If any pioneer engineers still work on the application, and they are proposing the change, the pioneers may need to get the rest of the organization on board with the investment needed. If on the other hand, in regard to the change, the request is not coming from the engineers then the engineers must be brought up to speed with why the proposed change is imminent and how it would benefit the organization or possibly the individual employees (Cummings et al., 2016; Hartzell, 2017). A standardize-able modernization strategy would factor the unfreezing stage of change to get all involved on board about whether and why a legacy application warrants modernization.

Changing. The actual change takes place in this phase, which is also known as the transitioning or moving stage (Hartzell, 2017). Once the change becomes a reality, everyone affected in the organization must learn new thought processes, general processes and behaviors to match the implemented changes (Cummings et al., 2016; Hartzell, 2017). For this reason, communication is also crucial in this phase to ensure that the change is maintained and that implications introduced by the change are addressed accordingly (Cummings et al., 2016; Hartzell, 2017). When thinking about the LSAs' lack of a modernization strategy, the organization would need a model to address this phase as well following a set of processes to ensure that the change completed smoothly. Some aspects to take into consideration include who does what change, at what time, and intervals as well as train all involved in the transition. A modernization strategy would also cover steps for the actual transition.

Re-freezing. This phase is also otherwise known as freezing as this is where the new changes are locked into place to prevent a rollback or reversal into the state before the change. At this step, the organization establishes a new norm (Cummings et al., 2016; Hartzell, 2017; Petiprin, 2016). According to Hartzell (2017), some scholars argue that this step may not be necessary if efforts are made to introduce a new organizational culture only to change it again when there might be a possible immediate need for change. However, according to Lewin’s model, this stage is for enforcing the change to ensure that the work in the changing stage was not in vain (Cummings et al., 2016; Hartzell, 2017). With regards to LSA modernization, this phase is also vital especially when the choice is to maintain the application, since if the maintenance strategy were not part of the organizational culture, then the change would have been in vain if the work should rollback.

The business problem addressed LSA modernization in this qualitative Delphi study in two parts, namely migrating and maintaining. While maintaining a legacy application is not necessarily a change in the application, there would be a change in the way of thinking about the application, once a set of processes is followed to determine that the system is not to be migrated to suit new technology, but merely maintained. Then, if the organization decides to migrate the legacy application to match a change in technology, the changes to be enforced may be more complex and demanding than maintaining the application. In either of the choices, the organization would have to decide on how the change would be applied and Lewin's model, being the foundation of change management is an excellent option on how the change can be applied.

Complexity Theory

Broadly known as the study of how designs, request, structure, and new themes emerge from clearly chaotic or complex systems and then again, how complex conduct and structure arise from basic and simple underlying rules (Cicmil et al., 2009). Complexity theory states that a system constitutes ongoing instability and entropy conditions, and as a result, varying structures and patterns emerge as the system evolves into something new (Lowell, 2016). Complexity theory stems from life sciences, physical sciences, and mathematics (Cicmil et al., 2009). Over 40 years, scientists made discoveries on nonlinearity and its effects on weather patterns varying drastically from one simulation run to the next (Kaufman, 1993; Lorenz, 1963). Biologists observed that organisms to adapt to life in different climatic conditions from those within which they evolved as a result of an effective response to predators (Kaufman, 1993; Lorenz, 1963). In doing so, characteristics and patterns emerge that are different in kind as well as in degree from the characteristics and patterns of the constituent components of the system (Kaufman, 1993; Lorenz, 1963). Complexity theory emerged as physicist took account of discrepant findings resulting from non-linear changes (Cicmil et al., 2009). Similarly, in the IT industry, the changes in technology have proven to be non-linear, as software and systems evolve in ways that were not predictable in the original systems (Cicmil et al., 2009). As a result, the old systems now lack information, are inflexible, lack IT resources, are dependent on specific individuals, and are costly to maintain and support; the applications are called LSAs (Stamford, 2014).

Complexity theory has many crossovers with the change management theory, and according to Lowell (2016), complexity theorists saw the study of non-linear dynamics in systems as a framework for reconciling why organizations with close to identical components produce unpredictable and divergent results with change management. In complex systems, massive changes may have little or no effect, while small changes may yield disproportional effects as a result of the unpredictable nature of interactions between individuals and the environment (Dutcher, 2011). According to Labs (2017), people should envision the world less as composed of machines but rather as composed of complex systems. Unlike machines that are a sum of their parts, functionality, and can be summed up as cause and effect, complex applications are non-reductive, non-linear, and have disproportionate cause-effect relationships, with new sets of models and methods of approach (Cicmil et al., 2009). Complex systems originate from mathematics, computer science, physics, and recently ecology, and can flip from one pattern of behavior to another but have no formal definition yet (Cicmil et al., 2009). Complexity is fundamental rather than amenable to the traditions of science, and as new patterns evolve in unpredictable scenarios, it becomes more difficult for project management to fulfill the challenges and requirements of increasingly complex projects and systems (Labs, 2014; Saynisch, 2010a, 2010b). The compatibility between an underlying infrastructure and the heterogeneous computing platforms-based interfaces, influence the complexity level of implementing and managing cloud computing heavily (Hwang, Huang, & Yang, 2016).

LSAs are large and complex custom-made software applications that are critical for business but resist modification, and their failure can have a significant impact on the business; or an application based on outdated technologies but critical to the organization’s day-to-day operations (Crotty & Horrocks, 2017; Srinivas et al., 2016). LSAs typically function as the backbone for critical daily business operations, making them indispensable to the organization (Rai et al., 2015; Srinivas et al., 2016). Therefore, it is vital to understand how complex systems work when finding a common standardize-able strategy for modernizing such complex systems. Another reason why the complexity theory drives this qualitative Delphi study is that even the definition of LSAs is complex and varies from use case to use case, so, before finding a common strategy, there should be a standard definition and understanding of what such systems entail (Beijert, 2016).

Disruptive Innovation Theory

Disruptive innovation occurs when a complicated or complex service or product is made less expensive and accessible to all so that it quickly and speedily moves up the market thereby displacing established competitors ("Disruptive innovations theory," n.d.). Though an innovation might be a breakthrough, it does not mean that it is disruptive, but disruptive innovations typically focus on consumers with no other option, must have an innovative business model, contain an enabling technology, and a coherent value network ("Disruptive innovations theory," n.d.). While some organizations avoid taking unnecessary risks by attempting to modernize their

LSAs, others want to match technology changes and be as competitive as possible while opening up their business to clients who do not have other options and the present study is basing on the latter set of organizations (Sandström et al., 2014).

This chapter presents the benefit of exploring the need for a common LSA modernization process while taking change management, disruptive innovation, and complexity theories into account. While it is likely that organizations have used different processes for modernizing their LSAs and may have even abandoned some, the literature reviewed in this chapter indicated the need for the present study as a step toward bridging the knowledge gap in this topic.

Legacy Software Applications (LSAs)

LSAs are large and complex existing application software, based on outdated technologies, lack documentation, resist modification, and are inflexible to meet changing business needs (Crotty & Horrocks, 2017; Srinivas et al., 2016; Stamford, 2014). As organizations gain prominence and technology changes, there should be a process in place to determine how to modernize the organizations' software applications (Islam et al., 2016). When initiating or planning new projects, the legacy status of a program is critical. LSAs play a prominent role in most organizations, and as such the organization may be in denial about the systems being legacy by using varying definitions for LSAs to avoid having to worry about them sooner than later (Beijert, 2016). Kaur et al. (2016) defined software complexity, legacy application, provided aspects of complexity, and the measure of challenge in modifying, analyzing, maintaining, testing or designing said software during the various phases of a software development lifecycle. Marcella and Rowley (2015) performed exploratory research to investigate the feasibility of efficiently applying software management tools and techniques across the creative industries. SMSEOs always have one project or another that affect a legacy application (Tantry et al., 2017). Being able to extend the life of these LSAs in organizations, would ensure that relevant organization's knowledge and processes are sustained (Tantry et al., 2017).

Background, Business Problem, and Significance

According to several studies, many IT organizations today lack a common strategy for modernizing their indispensable LSAs amid rapid technological and business changes, causing the ad hoc modernization efforts to be inefficient and costly (Crotty & Horrocks, 2017; Morton et al., 2015). Many organizations rely heavily on custom business applications that implement their unique business-critical functions and processes (Rai et al., 2015). These applications become part of the pool of LSAs that have to be maintained by the organization over time. Organizations still tend to have projects based on such systems, such as updating or removing existing functionality, and adding new functionality, and the older or larger the applications get, there is a need for a strategy to determine how to move forward (Islam et al., 2016).

Marcella and Rowley (2015) performed exploratory qualitative research, using a case study technique to investigate the feasibility of efficiently applying software management tools and techniques across the creative industries. Case study research suitable for describing multiple perspectives of a single subject like an organization using structured-interviews (Cooper & Schindler, 2014). Marcella and Rowley (2015) identified the importance of flexibility, lessons learned, and reflection were vital to modeling a successful project. Marcella and Rowley (2015), concluded that the values of project management tools and techniques could be demonstrated and adapted in creative industries as long as success criteria are accurately defined, and successful projects are delivered. According to Marcella and Rowley (2015), IT leaders lack a common strategy to modernize LSAs in organizations, and this is a business IT problem. Being able to apply existing software management tools and techniques across such organizations, would promote successful and cost-effective projects which relate to the rationale of the present study (Marcella & Rowley, 2015). Since modernization is also a type of project, applying management tools could be considered a modernization strategy, thereby narrowing down the options for a common strategy.

Rai et al. (2015) attempted to strengthen further academic research on modernization by migrating LSAs to cloud computing. Also, applied a systematic literature review (SLR) methodology to systematically and scientifically identify, compare, and categorize existing literature on cloud adoption (Rai et al., 2015). The results indicated that most of the reviewed literature focused on analyzing the requirements for planning and executing cloud migration and less on testing maintaining and monitoring. Rai et al. (2015) concluded that while cloud migration is evolving with time, there is a dire need for a secure migration model to increase organizational trust in cloud computing. The study by Rai et al. (2015) ties into the present research as a rationale for why a standardized strategy or model for legacy application modernization is needed, which would ideally fortify organizations' trust in cloud computing or any other migration destination. The findings of Rai et al. (2015) highlighted why migration is a viable option of modernization as well as the prevalent concerns about the reliability and trust for existing modernization strategies.

Dedeke (2012) defined a legacy application as a software package that has a technology, code, and standards from a past innovation, and loosely mentioned the time frame of 10 to 20 years. The author used an exploratory qualitative method to describe, explain, and determine the value of an LSA and how to extend, assess risk, analyze the portfolio or leverage organizational ethics. LSAs are complex systems and modernizing them is not a mere technological issue; additionally, multiple social, business, and organizational considerations exist as well, including but not limited to the reliability of the new technology or the decision-makers getting over their uncertainties (Dedeke, 2012). Dedeke's conclusion was to use ethical code or apply the proposed portfolio-based sustainability approach, which measured the value of an LSA according to the maintenance cost, service reliability, degradation factor, and quality of features. The author's proposal also included measuring the business value, by checking the system's competitive advantage, profitability, growth potential, inter-dependability as well as the weighted business value score. Dedeke (2012) showed that IT leaders and decision-makers care most about a standardized migration strategy and supports the background and rationale of the present study. Dedeke (2012) highlighted that a resolution to the legacy application migration problem would expose underlying ethical issues. According to Dedeke (2012), the lack of a common strategy implies IT managers must make critical decisions sooner or later to handle LSAs.

Islam et al. (2016) used an experimental quantitative study to predict LSAs migration from procedural to object-oriented paradigms. Used tools such as weighted data cell graphs

(WDCG), entity mapping, and weighted distance mapping to interpret the results of the experiment. Islam et al. (2016) then introduced new concepts supporting their proposal of a three-step migration technique which proved to outperform existing techniques. The three steps are WDCG generation from a procedural program, use agglomerative clustering to build a hierarchical cluster tree, and define an objective along with the level of the hierarchy that maximizes the function's objective (Islam et al., 2016). Used the Jaccard Similarity Coefficient to compare their results to those of other software engineers. Islam et al. (2016) concluded that their proposed solution provided the highest similarity when applied to five different C programs, as compared to the similarities provided by the other techniques. The conclusion showed that there is a need to find a standardize-able legacy application modernization strategy, which ties in with the present study (Islam et al., 2016).

Kaur et al. (2016) researched the lack of standardized processes for estimating the complexity of LSAs to be able to measure the decrement as well. The article explained the value of having a collective agreement on how to measure the complexity of a software application using a quantitative study to measure the complexity of various existing software. Also, Kaur et al. (2016) proposed the use of software complexity measurement as a part of the considerations of what action to take on LSAs that should occur at the beginning of a new project. The authors proposed a metric framework to use for the estimation and used the collected data to find correlations between security and complexity (Kaur et al., 2016). The article concluded that more methods and programs need to be applied to enhance the accuracy of the proposed estimation framework. Also, Kaur et al. (2016) provided some insight as to the variations of complexity in application software which further strengthen the need for a standardized strategy with support for complexity and change management theories all related to the present study.

Kehr, Quiñones, Böddeker, and Schäfer (2015) used a quantitative experimental research design to explore two challenges, namely extracting parallelism from a software application and introducing predictability in the functional behavior of the software. The authors proposed a timed implicit communication (TIC) strategy to allow parallel execution by decoupling task communication. TIC is a parallel implementation of automotive open system architecture (AUTOSAR) LSAs, with guaranteed data reproducibility and predictability on any multi-core electronic control unit (MCE), independent from the application workload with no change in the application source code (Kehr et al., 2015). The result of the experiment by Kehr et al. (2015) indicated that the strategy introduced reproducibility and predictability to the application software enabling the legacy application to execute tasks in parallel without code modification thereby introducing application sustainability.

In this OnBase (n.d.) White Paper, the author, addressed the management dilemma where LSAs stand in the way of insurers' agencies achieving agility, efficiency, innovation. The author uses a quantitative approach to analyzing secondary data from five sources, prepares a graph showing risks to benefits relation of five options for handling LSAs, and recommends a way forward. The five options were to use enterprise content management (ECM), buy a component, build a solution, replace completely, or do nothing all listed in order of increased risk and reduced benefits. The author recommends adopting the ECM strategy as it has the lowest risk and highest benefit. This white paper would help provide some context on the size of the impact of the legacy application lack of strategy issue since it states that about 70% of organizational applications are LSAs.

Radhakrishnan, Rouson, Morris, Shende, and Kassinos (2015) used a quantitative experimental design to demonstrate a strategy where Fortran77 codes are parallelized using the object-oriented (OO) Fortran 2008 co-arrays. The purpose of parallelizing the codes using OO programming is to be able to take advantage of OO features and therefore introduce extensibility to the legacy application initially written in Fortran 77. Another aim was to be able to modernize the legacy application, improve memory usage, and make it scalable. The results of the experiment showed an expansion of the original Fortran 77 codes with better load balancing but poor scaling as a recommendation for future studies. Radhakrishnan et al. (2015) reviewed test-driven development (TDD) as a strategy for legacy application modernization.

Srinivas et al. (2016) used a quantitative comparative study comparing various LSA analysis strategies, to help resolve the dilemma of being able to keep up with technological change pace and a possible update of business rules. The authors identified slow performance, costly maintenance, and the evolution of the applications, as some of the challenges of LSAs. Based on the comparison of 10 different analysis, the conclusion was that a common effective strategy is still needed, even though there were similarities in the strategies analyzed (Srinivas et al., 2016). This article supports the purpose of the present qualitative research and provides insight into the size of the impact of legacy application lack of a common modernization strategy.

Modernization of Legacy Software Application (LSA)

Organizational business need change is not always linear, so, the complex nature of such changes renders indispensable software application modernization complex, and failure could result in catastrophic consequences (Zheng, 2013). Modernizing a software application is a set of both managerial and technical activities and would make the application not only reusable but also more valuable, profitable, and reliable (Zheng, 2013). Some of the strategies in existing literature on software modernization include migration, maintenance, re-hosting as virtual machines, re-hosting on new hardware, replacement with new development, replacement with commercial-off-the-shelf (COTS), reengineering, wrapping, source code translation, retargeting, revamping, and evolution (Kuipers, 2002; Tantry et al., 2017; Zheng, 2013). Table 1 shows a comparison of the different modernization strategies, including criteria such as the advantages, disadvantages, cost involved, and the success rate.

Maintenance

Software application modification after delivery to either correct (corrective maintenance), adapt (adaptive maintenance), improve (perfective maintenance), or detect and correct potential faults (preventive maintenance) (Zheng, 2013). Maintenance is a significant part of the software development lifecycle (SDLC) which focuses on optimizing the software, correcting errors or removing unwanted functionalities (Quezada, 2017; Software maintenance, n.d.; Software maintenance overview," n.d.). Some researchers consider maintenance a modernization strategy since the software can be improved or adapted to new technology

(Kuipers, 2002; Zheng, 2013), while others do not since it runs on same old technology (Tantry et al., 2017).

Evolution

A combination of forward-engineering and reverse-engineering to continuously reengineer software applications, by identifying the application's components and their relationships, restructure or refactor the components, and then use software engineering techniques on the software application (Zheng, 2013). While perfective maintenance improves maintainability and performance, it only occurs post-delivery (Zheng, 2013). Evolution, on the other hand, is continuous and can occur at different phases of the SDLC and can either focus on the phenomenon or the methods and tools for modernization (Zheng, 2013).

Migration

According to ("Migration" n.d.), LSA migration is the act of moving from one operating environment to another one, which is typically more flexible, evolvable, and better than the previous one, as a way of modernization. Migration also allows for consideration of newer technology such as cloud computing (Zheng, 2013). Migration can either be component-based (migrating components of an application singly) or system-based (the whole system at once) (Zheng, 2013).

Tantry et al. (2017) presented other modernization strategies in addition to maintenance, migration, and evolution, as well as compared and contrasted the different strategies. Even though modernization has been occurring, the approaches have been different depending on the application and the modernizers.

Table 1
A Comparison of Existing Modernization Strategies and Activities

Modernization Strategy	Cost involved	Time required to implement	Success rate	Advantages	Disadvantages
Re-hosting: As virtual machines	Nil, any working Desktop is enough	Immediate	High	* No change in the working environment. * Reduce the maintenance cost of Legacy software	*Stop-gap option only. If the software itself is legacy, then this option is invalid
Re-hosting: on new hardware	Cost of new and compatible hardware	Immediate	High	* Reduction in the cost of maintenance. * working environment does not change * Increase in performance due to new hardware	* Stop-gap option only. If the software itself is legacy, then this option is invalid * Software remains still Legacy
Replacement: with re-developing	High	Long time	High	* Uses new technology * New modules can be added	* User training is required * Good testing is a must * Running old & new systems parallel attract more resources
Replacement: with Commercial-off-the-shelf (COTS) products	High	Immediate	High	* Adds new technology to the organizations' software life. * No time wasted.	* Business rule changes. * Customer have to tune the working environment to the pre-developed software
Migration	Medium	Moderate	High	* Same working environment. * Cloud computing technology can be utilized	* Database remains same as that of Legacy.
Re-engineering	Medium	Long time	Medium	* Improvement in the working environment. * Introduces modern Technology	* If the system study is not proper, then failure. * The developer should be well versed with both old & new languages & platforms.
Wrapping	Low	Moderate	High	* Legacy components can be used on web technologies. * Heterogeneous distributed computing environment can be utilized	* Efficiency is not improved in terms of legacy applications. * Only stop-gap option.
Source code translation	Low	Moderate	Medium	*Same code can be transformed to new platform with little manual involvement	* System to be tested again for the correctness on the new platform. * Structure of Legacy code remains same in new language also.
Retargeting	Medium	Immediate	High	* Reduces recursive operational maintenance cost.	Cost of new hardware may be a burden

Modernization Strategy	Cost involved	Time required to implement	Success rate	Advantages	Disadvantages
Revamping	Low	Moderate	Low	* Legacy hardware maintenance cost reduced Improves the visibility and usability of system at Front-End	* Internally same Legacy software, so maintenance is difficult. * All the problems of the Legacy system continues.

Note. Adapted from Implications of legacy software system modernization - A survey in a changed scenario, by Tantry, H. S., Murulidhar, N. N., & Chandrasekaran, K., 2017, International Journal of Advanced Research in Computer Science, 8, p 1007. Copyright 2017 by International Journal of Advanced Research in Computer Science. Adapted with permission. This researcher adapted by aligning the table style to that of a doctoral dissertation.

Some modernization strategies focus on an experience report, the decision-making, the architecture, technical aspects, or tool simulation but lack a common systemic strategy (Tantry et al., 2017). Tantry et al. (2017) identified the challenges of software modernization, which include high costs, lengthy processes which could cause the modernization to be obsolete pre-delivery, as well as the need for the parallel running of the LSA and its replacement.

Tantry et al. (2017) concluded that there is no one-size-fits-all strategy, however, while strategies like wrapping or transformation are cost-effective, the migration strategy ensures the use of new features and technology while maintaining the old setting of the LSA. The conclusion about no one-size-fits-all strategy influences the purpose of the present research, to find the perspectives of IT leaders on a common modernization strategy.

Zheng (2013) presented maintenance and evolution as modernization strategies. Business changes, customer requirements, and external changes are some of the reasons for modernization and understanding the decomposition of the software is vital for modernization (Zheng, 2013). Zheng (2013) proposed a modernization as follows, use a functional approach for understanding the applications such as program slicing to understand smaller parts of the software, and data type approach such as software clustering to mine data and recognize patterns in the LSA. Next use cloud-oriented evolution on the application that has been programmatically transformed, and finally implement a cloud service integration (Zheng, 2013). Zheng (2013) concluded that the proposed algorithm might not satisfy all kinds of LSAs, which applies to the need for finding a common modernization strategy in the present study.

Ruan, Vyas, Liu, and Spear (2014) used a quantitative methodology with a descriptive research design to study the possibility of modernizing by transactionalizing LSAs using transactionalized memory. Transactionalize is the requirement of having similar and consistently good interactions between individuals and groups ("Transactionalize," n.d.). Per the authors, transactional memory (TM) is an efficient programming idiom that must have proper specifications and broken down into three categories of transaction declaration, function annotation, and exception support. Ruan et al. (2014) also described a quasi-experiment where Memcached, a type of memory, can be transactionalized by identifying locks, refactoring condition synchronization, applying the specifications maximally, making libraries safe, and clearing transactions of code. The authors also recommended collaboration between programmers and system developers to achieve maximum results in transactional programming. Ruan et al. (2014) promised to release their modified version of Memcached as open-source code. The authors also discussed the chances of unintended cos of serialization, and the effort required to avoid the cost as well as that the specifications they used in their study could be changed to improve transformation analysis and programmability. Ruan et al. (2014) provided some existing modernization processes for LSAs.

Sadeghi et al. (2015) covered the vulnerability of LSAs code-reuse, considering that at the creation of most of the LSAs typically backward compatibility and performance were the priority with little or no security considerations. The authors discussed the various types of code-reuse attacks and some ways to prevent or slow down the attacks. Sadeghi et al. (2015) confirmed that LSAs had non-trivial problems which necessitated research into modernizing them, especially in the face of possibilities of networking between applications, some of which could be malicious.

Saini, Mehmi, and Chahal (2016) used a quantitative empirical, experimental design, and fuzzy data mining algorithm to analyze the number of commits to open-source software code. The authors generated association rules by evaluating the regularity and existing trends in the evolution of open-source software. Saini et al. used two sets of data to generate the association rules, namely the training dataset and the remaining dataset. The authors conclude that the association rules generated allow for the analysis of trends and regularities in the software. The recommended future research by the authors is to consider the concept of fuzzy data mining algorithm for time series data, predict the number of monthly commits, and further to investigate on other various factors that could impact the number of commits. The findings of Saini et al. (2016) support the present research by providing the data source for how LSA evolution data can generate association rules.

Serrano et al. (2014) addressed the concerns that organizations face, with the need to adapt their application software to keep up with the fast changes in technology in recent years. The authors discussed service-oriented architecture (SOA) such as Web Services for building more flexible applications as well as SOAP and REST protocols. Serrano et al. (2014) used a comparative quantitative study to compare the various enterprise application integration methods and allude to the fact that the choice of any of the options relies heavily on the needs of the organization. Serrano et al. (2014) cited a prediction made by Gartner that by 2016 integration platform as a service (iPaaS) would be at 35% increase, though other experts thought that iPaaS would not replace web services. The article by Serrano et al. (2014) aligns with the significance and the impact of the lack of a common LSA modernization strategy gap.

She, Ryssel, Andersen, Wąsowski, and Czarnecki (2014) performed an experimental quantitative study to attempt to resolve the NP-hard problem of automatically synthesizing feature models from propositional constraints using their proposed algorithm against the binary decision diagrams (BDD)-based approach and a formal concept analysis (FCA). The authors reported a 10 to 1000-fold improvement in performance for their proposed algorithm over the BDD-based approach. She et al. (2014) concluded that their proposed algorithm is the first known efficient technique to be used in extracting dependencies from real systems, creating an opening for reverse engineering. The article by She et al. (2014) was useful in the present research to add to existing strategies for migrating LSAs by extracting dependencies and creating the possibility for reverse engineering.

Vogel-Heuser et al. (2015) used a quantitative exploratory data analysis to study the evolution of automated production systems and the cross-disciplinary challenge it poses to the life cycle of high complexity systems. Vogel-Heuser et al. (2015) discuss the survey by the authors from computer science and automation, then talk about the development process of automated production systems, the various types of evolution in the life cycle of the system and illustrate on some challenges in the systems. The authors then surveyed the evolution in different SDLC phases and the associated problems. Vogel-Heuser et al. (2015) point out the lack of understanding in the human perspective on evolution as well as obstacles in the modification and reuse of the systems. This qualitative Delphi study supports the importance of the present research as it further supports the need for having a common modernization strategy for any, and all LSAs.

Marfatia (2014) discussed the 36% increase in the number of organizations targeted by Advanced Persistence Threats (APT), the daily cost of such loss, and the role of LSAs in them. The author discussed the difference between software migration and maintenance costs and leaned preference toward repairing and maintaining LSAs. Marfatia (2014) also suggested some steps that could be leveraged to maintain LSAs.

The present study explores the possibility of a common modernization strategy for LSAs, including the challenges that SMSEOs are facing in coming up with a strategy. Having a common strategy in place is beneficial to the decision-makers of SMSEOs because they would be able to continually provide services to both internal clients and paying customers while also leveraging modern technology as it changes. Also, it provides a structure to use for training and knowledge transfer to new hires as needed (Dedeke, 2012). Another additional contribution to scientific research would be a validation of the effectiveness of the research design and method used by applying it to this qualitative Delphi study. The study results also provide a guide for IT managers in the software engineering industry to determine when to completely replace banking software applications when as they become legacy and cost of maintenance becomes higher than the replacement cost. This research also helps regularize the general performance expectation from the LSAs, as well as budgeting, by helping with the existing problems with LSAs such as lack of flexibility, lack of information, lack of resources, and high support and maintenance costs (Stamford, 2014).

Deciding How to Modernize

Beijert (2016) used a qualitative study that explored and expanded on the high cost and risk involved in migrating LSA, resulting in the reluctance of organizational leaders to take any action about the LSA problem. According to Beijert (2016), many organizational leaders are reluctant to address problems with LSA resulting in application software coming to the end-of-life status. The author also pointed out how ineffective it is to maintain LSAs and the issues with both migrating and maintaining, which can lead to the end-of-life of the systems. Beijert (2016) proposed the definition of LSAs to be an application that limits or hinders organizational flexibility, growth, or innovation. The author also suggested a framework be used to designate an application as an LSA and therefore properly manage the expectation from a given system depending on its legacy status, all to promote future-oriented perspectives and response time. The framework classified the system under both the flexibility to change and business alignment where a combination of low change flexibility and low business alignment indicate that the system is an LSA (Beijert, 2016). This qualitative Delphi study contributes to the IT knowledge base in that LSAs definition would be standardized and accurately set the expectations from an LSA by first realizing or acknowledging that the system is legacy (Beijert, 2016).

EL Beggar, Bousetta, and Gadi (2014) addressed the problem in which decision-makers in organizations lack strategies for maintaining legacy COBOL systems and avoiding costly and risky system replacements as company activities and policies change. The author used a qualitative case study on a COBOL system by proposing a possible approach of reengineering and extracting business rules from LSAs. The design was to first obtain the business rule from the LSA and model the business rules by linking related ones. Next, obtain object methods from the extracted rules, formulate business rules, and correlate them. The authors concluded that their proposed business rule extraction (BRE) proposal provided a bridge between business logic and application code protecting the platform from technology rupture and reducing reengineering time. EL Beggar et al. (2014) support the impact of the present research as it aligns with the LSAs maintenance aspect of this qualitative Delphi study.

Hakemi et al. (2014) researched the risk analysis strategies for determining when to migrate LSAs. The authors used a quantitative comparative study between four risk analysis method namely CRAM, CORAS, OCTAVE, and VECTOR, and questionnaires for the data of the VECTOR method (Hakemi et al., 2014). OCTAVE is a mixed method in two phases, where the first phase is a qualitative interview to understand employees’ views on and inform them of the need for protecting data, uncover potential cases of data loss (Hakemi et al., 2014). The second phase is a quantitative survey based on data from phase-one. Hakemi et al. (2014) chose the combination of the two phases as the best LSA analysis strategy, but they listed no data collection instrument. The research by Hakemi et al. (2014) aligns with the present study by providing backing for the impact of the business problem, showing why it is essential for an organization to quickly analyze LSAs and determine whether to modernize by migrating or maintaining the systems.

Hwang et al. (2016) used a mixed research methodology to identify the context in which Science and Technology (S&T) institutions adopted cloud computing. The authors used a Delphi qualitative focus group approach to establish a set of determinants for using cloud computing. Then the cause-effect relationships were determined using a quantitative empirical survey DEMATEL analysis design. The results of the study revealed a range of challenges and uncertainties, preventing a widespread cloud computing adoption. The study by Hwang et al. (2016) supports the present study because they presented a strategy for modernizing legacy application by migration to the cloud.

According to Foster (2011), when the source code is unavailable, the organization is unable to recompile a software application to another stage. In such cases, some organizations resort to extraordinary measures, either purchasing the remaining parts of old machines to tear up (which postpones the inescapable) or building stage emulators to run the original code unaltered (Foster, 2011). Another alternative is to revamp the heritage code entirely in light of the fact that the organization cannot bear to keep a current code base around on the grounds that it is excessively costly, therefore impossible maintain, sustain, and is overly fragile, making it impossible to change (Foster, 2011; Jain & Chana, 2015). A reasonably sensible response to this situation is to have a project team modify the framework without any preparation (Foster, 2011). In the best-case scenario, this brings a much-needed update to a generally weak code base, in an engaging way; although copying an entire system in operation is unrealistic (Foster, 2011). One may be tempted to resolve apparent bugs in the code base. However, the modernization team most likely uncovers the fact that a few clients' (people or different frameworks) semantics dependencies (Foster, 2011). Therefore, a great deal of time can be spent finishing the framework's full formal operational semantics and figuring out how to manage delicately progressing the client base. Lavelle and Webb alluded that an organization may choose to maintain the legacy application over migrating it (Foster, 2011).

An old bit of programming may be creaky and fragile, yet its dependability has been inalienably reliable from long periods of utilization (Jain & Chana, 2015). The most widely recognized issue of sufficient programming is incorporating it with more current technology (Jain & Chana, 2015). An innovative impedance befuddles regularly exists between more seasoned centralized computer driven programming and more current frameworks (Jain & Chana, 2015). A way to deal with settling this issue includes advances, for example, SOA (Razavian & Lago, 2015). Organizations may change their legacy code, which is the most laborious activity since it requires a consistent hand, a strong vision for the future, and a deliberateness that is regularly absent within established organizations. This view is just refactoring, an emphatically demonstrated practice (Foster, 2011).

In essence, change becomes inescapable when a business-critical legacy chunk of code becomes hard to maintain, meaning the legacy code must be persistently changed (Jain & Chana, 2015). The expectation of change requires an effort since the need for dependability without hesitation, restricts persistent change (Foster, 2011). Such change requires a relentless vision since somebody must watch out for the skyline, driving the organization to where it should be in the near and far future, while the team frantically paddles toward the ultimate goal of modernization (Foster, 2011). Ultimately there is no one size fit all when it comes to deciding whether to modernize a legacy application by migration or maintenance as it all boils down to the organizations' needs.

Research Methodology and Technique

The Delphi technique is an iterative procedure used to gather and distill the judgments of specialists utilizing a progression of surveys blended with criticism (Skulmoski et al., 2007). The questions are intended to center on issues, openings, arrangements, or conjectures. With each ensuing phase created in light of the aftereffects of the last round. The procedure stops when an answer to the research question emerges, for instance, achieving an agreement implies an accomplishment of hypothetical immersion, or when adequate data has been traded (Skulmoski et al., 2007). According to Skulmoski et al. (2007), the Delphi qualitative technique has its causes in the American business group and has since been broadly acknowledged all through the world in numerous industry parts including social insurance, safeguard, business, instruction, data innovation, transportation, and design. The Delphi strategy's adaptability is evident in the way it has been utilized in qualitative research (Skulmoski et al., 2007). It is a strategy for organizing an amass correspondence procedure to encourage aggregate critical thinking and to structure models (Linstone & Turoff, 2011). The Delphi technique can also be used to gather the views of both insiders and outsiders of a given project (Linstone & Turoff, 2011). The technique can likewise be a judgment, choice supporting, or estimating device (Rowe & Wright, 1999). The Delphi technique is useful when there is inadequate literature on a given issue as per (Adler

& Ziglio, 1996). The technique can connect to issues that do not lend themselves to exact diagnostic strategies yet instead could profit by the subjective judgments of people on an aggregate premise and to concentrate their aggregate human insight on the present issue (Linstone & Turoff, 2011).

Additionally, the Delphi technique is utilized to examine what does not yet exist (Skulmoski et al., 2007). The Delphi technique is a developed and extremely versatile research technique utilized as a part of numerous exploration fields by specialists across the globe. To better comprehend its decent variety in the application, one needs to consider the causes of the Delphi technique. This research approach is guided by the exploratory questions to satisfy a target of the present investigation (Cooper & Schindler, 2014). The Delphi technique was best for this qualitative Delphi study because, a subjective technique is appropriate when attempting to deduce a more profound comprehension of a circumstance (Cooper & Schindler, 2014). This approach offers a means of understanding the participants' perspectives and how their experiences contribute to their perspectives (Creswell, 1998). The qualitative information gathered utilizing a multi-round Delphi study, offers a full comprehension of rich, logical, and meticulous information (Cooper & Schindler, 2014). In this approach, scientist endeavor to decipher the importance of individual reactions (Creswell, 1998).

Skulmoski et al. (2007) depicted the Delphi strategy as the most fitting adaptable system when there is fragmented information about the exploration wonders. The authors recognized that this methodology is exhaustive and perfect to use because it is a developed strategy for researchers attempting to explore a problem. Further, Skulmoski et al. (2007) illuminated the

Delphi technique demands unsurprising responses from a chosen group of experts instead of from individuals picked arbitrarily.

Linstone and Turoff (2011) described the Delphi as a collective hierarchical and group arranging framework for organizing a gathering correspondence and choice process. The Delphi technique encourages the procedure as a reliable method to permit a panel of specialists to discuss a complicated concept or problem. Furthermore, some recognized advantages to Delphi include a controlled discussion in which individuals can participate non-concurrently and not be affected by any coincidental prompts surrounded by the analyst. Chapter 3 covers the criteria for selecting the board of IT leaders using the online web-based social networking system such as LinkedIn, Google Forms, and emails.

The Delphi technique requires the information gathering process to work in numerous rounds until the point that no new data is picked up from resulting rounds, and a level of the accord is achieved (Linstone & Turoff, 2011). According to Maxey and Kezar (2016), the sample size for a conventional Delphi technique research ranges from 30 to 60 participants. Fletcher and Marchildon (2014) used 39 participants in their research using the Delphi technique. Per "Populations and sampling" (n.d.), explorative qualitative studies need a relatively smaller sample size, and with the already mentioned considerations. McMillan et al. (2014) recommended a participant size of no more than seven after testing with groups of between two and 14 participants using the nominal technique. Skulmoski et al. (2007) posited that many researchers attain saturation using between 10 and 15 participants for Delphi studies, and also, McMillan, King, and Tully (2016); Shah (2017) used a sample size of 12 participants in the Delphi technique dissertation.

The aspects to explore are their reactions to LSAs, their opinions on how to approach the migration or maintenance decision, and how they are currently trying to resolve the problem. Similar to El-Gazzar et al. (2016), who studied the adoption of cloud computing as a means to resolve issues with LSAs, the present researcher planned for the data collection in three rounds, between the Policy Delphi technique and the ranking-type Delphi technique. First a brainstorming session on the available approaches for deciding how to modernize, whether to migrate or maintain LSAs, then narrowing down the outcome of the first round to find common themes and lastly using a moderated ranking technique to determine a standardized list of there is a consensus. Given no convergence, or if further questions arose, conduct additional session regardless if the same people attend or not, with the hope that even if same people attend, there would be the change in time factor that may or may not influence the emerged questions and theories.

Per (Cooper & Schindler, 2014), missing data occurs when one or more research variables lack information or response from one or more participants, either because they drop out of the study or do not know or provide the response to some question(s). The existence of missing data can lead to a biased result for the study, and if a researcher only chooses complete data from the responses there is a risk of sample size reduction; therefore, missing data must be handled with care (Leedy & Ormrod, 2014). To handle missing data, first, the data should be studied for patterns, missing data type, and then a technique chosen for dealing with the deficiency while avoiding bias (Cooper & Schindler, 2014). The three types of missing data include missing completely at random (MCAR), missing at random (MAR), and not missing at random (NMAR) (Cooper & Schindler, 2014). Also, per (Baruch & Holtom, 2008), electronic data collection results in a higher response rate, so the brainstorming round of data collection was executed electronically to increase the response rate and ensure there is enough left to work with after dropping the missing data responses.

Hwang et al. (2016) used a Delphi technique in a qualitative focus group methodology. The Delphi technique allows a group of experts to deal with a problem in a structured communication setting, such as a conference workshop (Maxey & Kezar, 2016). Hwang et al. (2016) used a focus group to collect rich descriptive data from four to twelve knowledgeable participants or subject matter experts in a two-phased format. First, the authors applied the DEMATEL analysis design, which is typically used to determine if solving one problem can help resolve another. The authors subsequently used the PLS-SEM formats to uncover the extent of cause and effect relationship between the research determinants (Hwang et al., 2016).

Literature Supporting the Sampling

In a typical Delphi study, the authors used a focus group to collect rich descriptive data from four to twelve knowledgeable participants or subject matter experts (Hwang et al., 2016). Hwang et al. (2016) used a two-phased format. First, they applied the DEMATEL analysis design, which is typically used to determine if solving one problem can help resolve another.

Islam et al. (2016), experimented using five C programs, in which they used tools such as weighted data cell graphs, entity mapping, and weighted distance mapping to interpret the results of the experiment. Also, they introduced new concepts supporting their proposal of a three-step migration technique, which proved to outperform existing techniques. The authors used the Jaccard Similarity Coefficient to compare their results to those of other software engineers.

Kaur et al. (2016) suggested that the metrics of complexity could be chosen from the lines of code (LOC), Halstead method, McCabe complexity or cognitive weight method, and applied their recommended metrics to existing LSAs. Per Marcella and Rowley (2015), because their topic had not been researched in-line with social science before, they chose an exploratory qualitative study. Rai et al. (2015) applied the SLR methodology to study 30 papers between 2009 and 2014 to understand migrating to cloud computing and security nuances better. The authors used existing data, which in this case were automated production systems of high complexity systems (Vogel-Heuser et al., 2015). The authors also discussed the cross-disciplinary challenge that automation poses to the life cycle of high complexity systems because such systems are variant-rich and that the problem space variable may differ from the solution space variable because of differing views of the developers (Vogel-Heuser et al., 2015).

Summary

To summarize, modernizing large indispensable LSAs is challenging. Research on the topic of software modernization identifies several vital influencing factors that contribute to its complexity. According to Kaur et al. (2016), while standardized software complexity measurement technologies exist, there is a lack of agreement on how to apply software complexity measurement in software modernization strategies. Additional considerations include organizational trust of the new technology to migrate to, as well as the assessment of the need, process, and cost of modernization (Crotty & Horrocks, 2017). Finally, the lack of a common strategy to enable efficient and effective modernization of software application is yet another critical challenge (Vogel-Heuser et al., 2015). This Literature review has presented support for the existence and relevance of the current technical business problem regarding the lack of a common modernization strategy for LSAs from the perspectives of disruptive innovation, change management, and complexity theories.

CHAPTER 3. METHODOLOGY

Introduction

Chapter 3 covers the design methodology, technique, participants, setting, analysis of research questions, credibility and dependability, plans for data collection, data analysis, and the ethical considerations of this qualitative Delphi study which are, critical aspects of qualitative scholarly research. The business IT problem is on indispensable LSAs which are complex applications that are the backbone for daily business-critical operations of an organization, and though typically under-documented and hard to update, the systems are also hard to maintain or replace (Rai et al., 2015; Srinivas et al., 2016). According to Srinivas et al. (2016), organizations need a Business Process Reengineering (BPR), to modernize LSAs and the LSAs typically pose some challenges to the organizations such as slow speed and high maintenance costs as well as high-cost fault detection typically due to obsolete technology. Similar to El-Gazzar et al. (2016), the research technique for this qualitative Delphi study is a ranking-type multi-round Delphi technique, using ideally three rounds, namely brainstorming, narrowing down, and ranking.

Design and Methodology

A research design is an outline or blueprint showing how aspects and variables are studied and plans for enforcing ethical standards and fulfilling the research objective; the design in this qualitative Delphi study is exploratory (Cooper & Schindler, 2014). The research design used was an exploratory qualitative methodology with a Delphi technique. A qualitative methodology is utilized for studying the meaning rather than frequency of specific phenomena by translating, describing, decoding, or exploring participants' viewpoints using interpretative techniques (Cooper & Schindler, 2014; Roberts, 2010). A research technique is a method of obtaining information using a logical approach and confirmation of a study's feasibility (Cooper & Schindler, 2014). A Delphi technique is intended for allowing a group of experts to deal with a problem within their expertise domain, in a structured communication setting such as conference workshops or teleconference meeting sessions, to arrive at a consensus (Hsu & Sandford, 2007).

The Delphi technique is a controlled investigative exercise to gather reliable data from the conclusions of an objective gathering of specialists (Maxey & Kezar, 2016). Linstone and Turoff (2011) described the Delphi technique as a collective hierarchical and group arranging framework for organizing and gathering correspondence and choices. The Delphi technique is an iterative procedure that can incorporate busy and autonomous specialists who were most likely be unable to meet face to face (Maxey & Kezar, 2016). It enables the individuals to conceptualize and discuss the information gathered utilizing mixed surveys (Shah, 2017; Skulmoski et al., 2007). Using the Delphi technique is appropriate for integrating feedback from multiple experts in multiple stages (Beijert, 2016; Skulmoski et al., 2007). Some advantages to using the Delphi technique include:

- It enables people to non-concurrently participate in the examination of a problematic issue at any given moment (Maxey & Kezar, 2016).
- It encourages a panel of specialists to discuss a complicated concept or problem.
- Participants are not affected by any coincidental prompts by the analyst.

- The Delphi technique allows for the information gathering process in multiple rounds until the point when no new information from resulting rounds and a level of convergence is achieved (Linstone & Turoff, 2011).

RAND Enterprise initially explained the Delphi method in the 1950s, to build up a shared method to connect with a board of specialists in an efficient procedure of reaching a consensus (Maxey & Kezar, 2016). The Delphi method is most suitable for leading a study when there is incomplete information about an issue or phenomena (Skulmoski et al., 2007). Therefore, the Delphi technique would help to mainly gather information on the complex topic from a group of specialists. The present research is for identifying a common strategy for modernizing LSAs by analyzing the perspectives of a panel of experts to attain a convergence; therefore, Delphi is a suitable technique for this qualitative Delphi study.

Participants

This section covers the participants, including the recruitment, selection process, saturation, inclusion criteria, as well as exclusion criteria, and the justification for each. All aspects were backed using references from other researchers who have successfully conducted scientific research, especially using the Delphi technique. In a Delphi study, choosing the right participants is crucial since the results of the studies are experience-based on the experiences and opinions of the experts, although there tends to be a wide range regarding the sample size of such studies (Shah, 2017; Skulmoski et al., 2007). According to Mason (2010), the mean sample size for qualitative Ph.D. studies is 31. Per Maxey and Kezar (2016), the sample size for a conventional Delphi technique research ranges from 30 to 60 participants. Fletcher and

Marchildon (2014) used 39 participants in their study using the Delphi technique. McMillan et al. (2014) recommended a participant size of no more than seven after testing with groups between two and 14 participants using the nominal technique. Per Cooper and Schindler (2014), explorative qualitative studies need a relatively smaller sample size to attain convergence. McMillan et al. (2016) and Shah (2017) used a sample size of 12 participants in their Delphi technique dissertation, that is why the present researcher is confident that the Delphi technique with a sample size of ideally 12 participants is a suitable method for the present research.

Population and Sample Size

For this qualitative Delphi study, the population was made up of IT leaders, including team leads, IT managers, IT directors, CTOs, CIOs and other decision-makers of SMSEOs, who have modernized or were currently modernizing one or more LSAs (Shah, 2017). Per Cooper and Schindler (2014), the sample frame describes elements that involve the target population from which a sample gets drawn. The sample-frame was IT leaders who were currently in the process of modernizing legacy software or had done so in the past and had at least five years of leadership experience (Shah, 2017; Skulmoski et al., 2007). Since the qualitative method a nonprobability methodology, the sample size depends on saturation, which refers to the point when no new themes or information can be observed (Guest, Bunce, & Johnson, 1995). Most literature using the qualitative method, show that a researcher can achieve data saturation with a small number of participants and some Delphi researches have used 12 participants (McMillan et al., 2016; Shah, 2017). Therefore, the present researcher determined the sample size for this qualitative Delphi study as ideally 12 IT leaders that have or are currently dealing with indispensable LSAs.

Saturation

According to Skulmoski et al. (2007), looking back at a variety of Delphi studies in a four-decade period, there is no set rule on how to determine the sample size for a Delphi study. Some guidelines in determining the sample size include heterogeneous vs. homogeneous samples in which the homogeneous group is smaller and may yield passable results while the heterogeneous is more massive and could render it harder to collect data or even come to a consensus. Also, although verifying data from a larger group may be more convincing, it is nevertheless challenging to analyze and prone to group errors, but there is a tradeoff on quality with the exclusion of larger sample in favor of smaller sample size (Shah, 2017; Skulmoski et al., 2007).

Skulmoski et al. (2007) posit that many researchers recommend the saturation of participants as10 to 15 for Delphi studies, hence the decision to use a sample size of up to 12 participants (Bowen, 2017; McMillan et al., 2016; Shah, 2017). The goal was to solicit about 20 participants so that even if some fall off, there should still be at ideally about 12 participants. Although there could be a tradeoff on quality by excluding larger sample sizes, with this smaller pool, it would be easier to analyze and manage the data collected (Bowen, 2017; Shah, 2017).

Recruiting Criteria and Procedure

This researcher created a LinkedIn group, and Google Forms explicitly for this research. The Google Forms included an introduction, a pre-participation/recruitment questionnaire, and a form per Delphi round. In the solicitation message, the potential participants were asked to participate in a multi-round Delphi study as well as forward the link to potential experts that they know, for snowball recruiting (Bergner & Lohmann, 2014; Shah, 2017). The recruitment questionnaire was sent via LinkedIn direct messages for recruiting the participants using purposive sampling setup to collect participants email addresses from their responses (Bergner & Lohmann, 2014; Shah, 2017). Purposive sampling is a technique where the participant choices are deliberate because of the quality they possess (Etikan, Musa, & Alkassim, 2016). The inclusion criteria demographics were: IT leaders that have successfully led a team of at least five people, in modernizing LSAs or was in the process of doing so (Shah, 2017; Skulmoski et al., 2007). Some job titles of participants included technical leaders, project manager, program manager, project leader, senior software engineer, senior applications developer, system analyst, software architect, senior test engineer, senior operations manager, CIO, and CTO.

Additionally, the participants should have worked in the SMSEO industry for at least five years, have essential communication skills, be reachable, be able to set aside sufficient time and be willing to participate in a multi-round Delphi study within a one-month time frame (Shah, 2017). The exclusion criteria included allowing only one participant per company to promote diversity (Bergner & Lohmann, 2014). The plan was for the present researcher to send Google Forms invitations to the potential participants via LinkedIn direct message (Shah, 2017). Then after receiving potential participants' responses to the pre-participation questionnaire via Google Forms, analyze them and send the Institutional Review Board (IRB)-approved informed consent via Google Forms to those who fit the inclusion criteria and not excluded by the exclusion criteria. The form also contained a question about their availability for teleconferencing, and the link to join the LinkedIn group where the present researcher planned to communicate next steps with all the participants (Shah, 2017). The recruitment form took the form of a brief introduction, and a demographics questionnaire (see Appendix B for a sample of the researcher-designed recruitment questionnaire and Round 1 question) (Shah, 2017). The potential participants had one week to respond to the questionnaires, and the present researcher planned to send a follow-up reminder if a potential participant did not reply within one week. Only potential participants that fit in the inclusion criteria and signed the consent form were allowed to participate in this qualitative Delphi study and excluded anyone unable to make the time to participate in the multi-rounds (Bowen, 2017). This researcher would send a thank-you email or LinkedIn message to those who did not fit the inclusion criteria, thanking them for their interest, and indicating the missing qualification (Bowen, 2017). The follow-up letter was sent thrice within a two-week time frame, and when the potential participant still did not reply, the present researcher assumed the potential participant had dropped or was not interested. Then, the present researcher sent the Google Forms link to the IRB approved informed consent, including a checkbox to consent or decline participation in the current Delphi study, to all who qualify to participate.

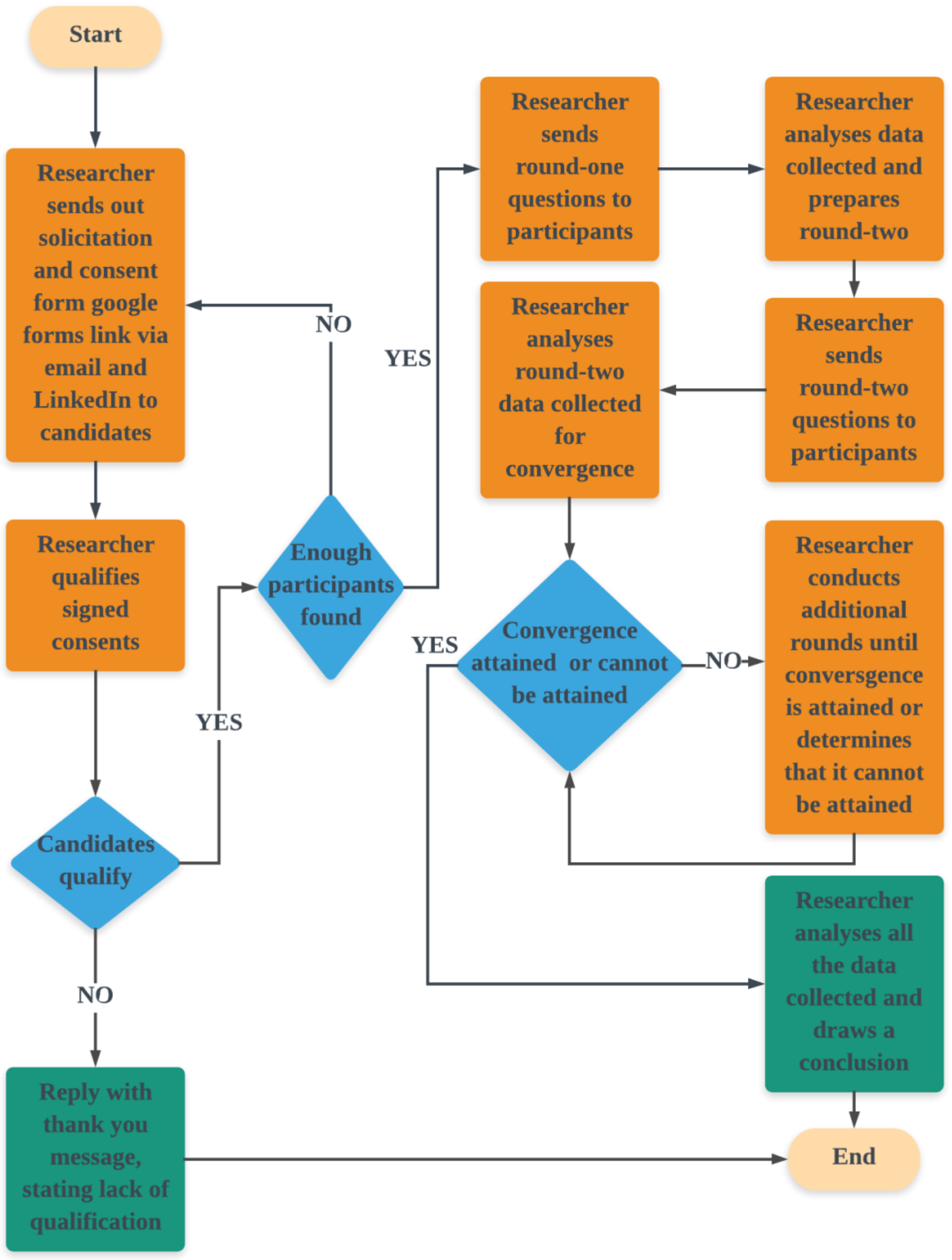
Setting

As seen in Figure 1, the present researcher planned to send the potential participants for participation a link to the solicitation letter, hosted in Google Forms, with details of the study via LinkedIn direct message. Also, the present researcher encouraged the potential participants to forward the link of the recruitment form to others who could be qualified and were interested.

The solicitation email contained a recruitment questionnaire, including demographics and background details, to help determine if the potential participant could participate. If after one week a potential participant had not replied, the present researcher would send a follow-up letter to gently nudge the potential participant to reply as indicated on the flow chart using the second blue arrow from the researcher to the potential participants. The follow-up letter would be sent up to three times via LinkedIn direct messages, email, or phone depending on the given stage and if the email or number was available. This researcher aimed to get at about 20 signed consents to participate within two weeks. Subsequent communication between the participants and the present researcher is indicated using the orange process fields in Figure 1 which includes instructions, one open-ended question per round via Google Forms, the LinkedIn group, email, or teleconference tools such as Adobe Connect, GlobalMeet, or Skype depending on the availability and time zones of the participants. The dark green process fields indicate finalizing, either with unqualified potential participants or concluding the data collection process.

All participants' responses were only visible to the present researcher and collected as follows:

1. Potential participants received a Google Form link via LinkedIn direct messages.
2. A thank-you message, encouraging the respondents to forward the link to others.
3. After a week of no response, the potential participant received one reminder every two days, up to three times, and removed from the potential participant list if no response.
4. Evaluated each response against the inclusion and exclusion criteria to determine if the potential participants qualified to participate.



Start
Researcher sends out solicitation and consent form google forms link via email and LinkedIn to candidates
Researcher qualifies signed consents
Candidates qualify
NO
Reply with thank you message, stating lack of qualification
YES
Enough participants found
NO
YES
Researcher sends round-one questions to participants
Researcher analyses data collected and prepares round-two
Researcher sends round-two questions to participants
Researcher analyses round-two data collected for convergence
Convergence attained or cannot be attained
NO
Researcher conducts additional rounds until conversgence is attained or determines that it cannot be attained
YES
Researcher analyses all the data collected and draws a conclusion
End

Figure 1. Research setting, solicitation, and recruitment of potential participants, and communication between the researcher and the participants during the Delphi rounds.

5. For those that qualified, the present researcher sent them the IRB approved informed consent via the email address provided in the recruitment questionnaire to sign and then join the LinkedIn group where the present researcher would share general information.
6. For those that did not qualify to participate or did not sign the informed consent form, the present researcher would send a message thanking them for considering, let them know the present researcher would not contact them further.
7. Once the present researcher had the desired number of signed informed consents, the present researcher would send a message on the LinkedIn group, letting everyone know that it is time for data collection to start. Then, share the link to the Round 1 question along with the participation guidelines such as deadlines, and that the present researcher would reach out to them individually with any questions regarding their responses.
8. This researcher would use the LinkedIn group for sharing general information about the participation procedures. If the participants had non-personal questions, they could also ask on the group, and the present researcher would respond there so that everyone knew the response as well. If the question turned out to be personal or, the answer was personal, and then the present researcher would reply to the participant privately.
9. If the present researcher were unclear about an individual's response, the present researcher would ask the individual via direct private LinkedIn message or email, for clarification.
10. Once the present researcher got all responses, or the deadline passed, and no further answers came even after reminders, the present researcher would let the participants all

know via the LinkedIn group and how long it would take for the present researcher to get back to them.

11. If this researcher’s data analysis warranted another round, the present researcher would let the participants know, present the outcome of the previous round without revealing who said what, and then provide the link to the next round via the LinkedIn group.
12. If this researcher’s data analysis showed convergence or a conclusion, then the present researcher would provide the participants with an outcome, request for their feedback on the outcome, and thank them for participating all via the LinkedIn group.
13. This researcher downloaded all the data collected and deleted them from the Google Forms page. Then coded the participants’ identity, for example, using “Participant 1”, “Participant 2” to protect their integrity, and maintained the key to codes in a separate document, in case the present researcher needed to contact them for clarification at a later point during data analysis.
14. Once the present researcher completed the analysis, the present researcher downloaded and stored all the data collected on a flash drive, in a locked cupboard in this researcher’s home.

Analysis of Research Questions

Typically qualitative studies involve questions and sub-questions, which all combine to answer the primary research question, and the questions are sometimes pre-tested to ensure validity and reliability of the research (Cooper & Schindler, 2014). This researcher wanted to have a synchronous Round 1 session if all the participants could be available, in which the participants would be invited to attend an audio-only recorded teleconference call meeting (Aengenheystera, Kerstin Cuhlsb, Heiskanen-Schüttlera, Huckd, & Muszynska, 2017). This researcher found no published instrument that addressed the adopting of a common LSA modernization strategy. As a result, the present researcher designed the questions for the recruitment questionnaire as well as for each round, which evolved depending on the consensus from the previous round (Skulmoski et al., 2007). Round 1 question was open-ended and subjective, so the present researcher could use the responses to determine the Round 2 question and the need for the subsequent rounds. If a synchronous meeting were not possible, then the participants would still have sufficient time to respond to the question without any rush. Therefore, the present researcher did not pre-test the questions. All communications about the next steps, timeline, and processes were communicated via email and direct LinkedIn messages (Shah, 2017; Skulmoski et al., 2007). The participants were given a deadline to submit their responses for each round (McMillan et al., 2016).

This researcher let the participants know that they could ask questions privately about the process, or if any question was unclear, as well as that this process would continue in more rounds until no new ideas came in (McMillan et al., 2016). In the asynchronous second session, the results from the first round were presented to the participants without disclosing the respondent, so as not to impact other participants responses because of a respondent, preferably by the actual data and how or whether they would consider it as an option now that they know that other experts consider it as well (Aengenheystera et al., 2017). This researcher closed each session with the prospect of having the next round as needed (Shah, 2017; Skulmoski et al., 2007). More rounds would be added as needed until no new ideas emerged, and convergence attained (McMillan et al., 2016).

The overarching research question was: What common modernization strategy can IT leaders in SMSEOs leverage to modernize their indispensable LSAs as technology changes? This researcher used a multi-round Delphi technique, to answer to this overall research question, by collecting and analyzing responses from each round to determine convergence and create the follow-up questions in the subsequent rounds. The sub-questions considered were:

Sub Question (SQ) 1. What business or technical imperatives drive the modernization of LSAs in SMSEOs?

SQ 2. What challenges impact the modernization process?

SQ 3. What factors contribute to the success of the modernization?

Credibility and Dependability

According to Guba and Lincoln (1989), conducting sound qualitative research requires the use of the four elements transferability, credibility, conformability, and dependability. Dependability is all about the context as well as the changes that may occur within the research (Shah, 2017). The participants mainly evaluated credibility by checking the researcher's interview transcript and referred to the reliability of the research results (Shah, 2017). For dependability and validity, the present researcher only considered the opinions of the subject matter experts (participants) as data for analysis (Bowen, 2017).

While the credibility of the research outcome was not assessable before realizing the results, a well thought out design during the research would most likely result in results that are credible. The Delphi technique is a controlled investigative exercise where one inquiry was solicited to gather dependable accord from conclusions from an objective gathering of IT security specialists (Maxey & Kezar, 2016). The Delphi technique uses a structured group communication method with a specially selected set of participants chosen for their expertise in the research area (Maxey & Kezar, 2016). The topic is well suited to a qualitative Delphi methodology as the intent is to explore perceptions of a common LSA modernization strategy (Skulmoski et al., 2007).

The recursive nature of the Delphi technique with three iterative rounds of inquiries presented to IT experts who are anonymous to one another allows for uninfluenced and unbiased information, increases the dependability and validity of the study (Deschene, 2016). Additionally, it was essential to recognize individual bias as the researcher, however, through such recognition, seek to analyze the ongoing research results from all angles, including one's predispositions (Deschene, 2016). Therefore, with robust research design, rigorous application of the plan, readiness to be open to different and new points of view concerning one's own, ensured that the results accomplished credibility (Deschene, 2016).

Data Collection

The goal of data collection in the present research was to find the challenges that IT leaders face in determining an LSA modernization strategy and then converge on one set of managerial and technical activities as a common strategy for LSA modernization (Hakemi et al., 2014; Serrano et al., 2014). Having procedures and policies in place is beneficial to the decision-makers of software engineering organizations because they would be able to continually provide services to both internal clients and paying customers while also leveraging modern technology as it changes (Bakhit, 2016; Vecchiato, 2017). Considering that the results of this qualitative Delphi study would affect many SMSEOs, the decision not to use a single organization came easily, since the variety of input and experience of experts from various firms would lead to much richer results.

This researcher sent LinkedIn direct messages and emails to the potential participants, which included this researcher's introduction, the topic of study and that the participation is voluntary (Bowen, 2017; Shah, 2017). The potential participants were made aware that the present qualitative study will be published, and the participants would have access to the publication; also that participants' names will remain undisclosed inside the published document, and no one besides the present researcher will have the participants' names (Shah, 2017).

The potential participants were additionally made aware that a physical copy of the data collected will be secured for at least seven years after the research was approved and then collect digitally signed consents from them to proceed (Shah, 2017). As part of the solicitation letter, the potential participants received a demographic survey, which besides the email, organization, and job title, includes questions to determine the potential participants' team size, length of time in a leadership role, availability for a teleconference call, as well as number of successful modernization projects and modernization challenges (Aengenheystera et al., 2017; Shah, 2017). The results from the initial survey determined the setting of the Delphi rounds, as in whether it would be synchronous or not and who could participate (Shah, 2017).

For this multi-round Delphi study, the first round included a more detailed introduction, the topic, and purpose of the present study, the selection criteria of the participants and why the research results stand to benefit them regardless of the outcome (Skulmoski et al., 2007). Also, the instructions contained the description of the round, structure of the questions, the definition of any ambiguous terms, and the next steps leading to rounds two, as well as letting them know of the chance of having Round 3 (Shah, 2017). The goal was to see the responses of the participants converge toward one consensus, so Round 2 included questions on the reason for the choice along with the pros and cons of choice, to determine if after knowing the strategies of the other experts, there was a change in opinion. If the Round 1 results consisted of all unique views from the different participants, then Round 2 would contain all the views to find convergence. If after Round 2 and Round 3 there was no convergence and no new information, then some conclusions would be made with the help of the demographic data collected earlier (Skulmoski et al., 2007). On the other hand, if Round 2 results immediately converged, then also use the demographic data to draw some conclusions (Skulmoski et al., 2007). Otherwise, perform subsequent rounds to get more convergence or until no new information emerged, while also collecting why participants change their opinions if they do (Shah, 2017; Skulmoski et al., 2007).

The aspects explored were participants reactions to LSAs, their opinions on how to approach the modernization decision, and how they had in the past or were currently trying to resolve the problem. Similar to El-Gazzar et al. (2016), the present researcher conducted the data collection three rounds using the ranking-type Delphi technique. The first round was a brainstorming session on the available LSAs modernization approaches and strategies (Bowen, 2017). Then, narrow down the outcome of the first round to find common themes, and lastly using a moderated ranking technique to determine a common strategy if there was a consensus (Aengenheystera et al., 2017). Given no convergence, or further questions arose, the present researcher would execute more rounds unless the round introduced no new information (Aengenheystera et al., 2017).

Data Analysis

The data collected was in the form of words typed format since the responses were free text. The data analysis involved coding by compiling the responses to find similar opinions, grouping similar findings together, and also highlighting the differences between the responses (Bowen, 2017). This researcher used a combination of constant comparison analysis (CCompA), and classical content analysis (CContA) which are very similar but for the fact that while both find themes in the data, CCompA accounts for the number of times an idea is used (Leech & Onwuegbuzie, 2007). This researcher will use CCompA to merge the unique ideas using common key phrases and words and compile a list for the participants to select from in the second round. Then after attaining convergence, the present researcher would use CContA to prioritize the data by popularity.

If after Round 2, no convergence had been reached a third round would be executed still containing the unpopular choices (Bowen, 2017). So, for the views that only one participant chooses, form a new questionnaire with those listed as possible common strategies for modernizing legacy application systems. The participants would then respond to those via yes or no answers if they would choose a given strategy and why (Shah, 2017; Skulmoski et al., 2007).

If there were still no convergence after the third-round, the present researcher would perform a fourth-round with the cumulative list of prevalent options to determine how popular they were and if a conclusion was possible. However, if convergence were reached after any round starting from the second round, then a next round would be to confirm on whether the participants agree to this researcher's conclusions or not (Shah, 2017; Skulmoski et al., 2007).

Ethical Considerations

This research accommodates and accounted for ethical and moral standards as per the Belmont Report. The Belmont Report covered three basic standards: (a) regard for the individual, (b) value, and (c) human subject research justice and guidance ("National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research," 1979). In alignment with the Belmont Report standards, the researcher provided an IRB approved informed consent form to be signed by each participant, expressing the nature, degree, and goal of the study; confidentiality and security affirmation; and the information gathering method (Bowen, 2017). Furthermore, the present researcher provided information on all the benefits and risks introduced by the research ahead of time to every one of the participants. This qualitative Delphi study was in full compliance with the IRB controls (as indicated in the school's conflict of interest and dual roles IRB guidance on understanding, identifying, and managing conflicts of interest). Also, the likelihood and level of danger of participants' distress or harm anticipated in this qualitative Delphi study were no more than one would experience in regular daily activities or amid the execution of routine physical or psychological tests (Bowen, 2017; Shah, 2017).

Furthermore, all responses from the participants will be stored on an external hard drive not available to the public but may become available if required by the IRB (Bowen, 2017; Creswell, 1998). The study poll utilized as a part of the present research excluded the participants' personal identification information, such as name, address, and organization data. This way, their anonymity, privacy, and security remained protected. The data collected was fully compliant with industry standards, privacy, regulations, and security policies (as indicated in the school's conflict of interest and dual roles IRB guidance on understanding, identifying, and managing conflicts of interest).

Additionally, the present researcher only used the participants' contact information, such as email addresses for maintaining communication with the participants during the data collection process (Bowen, 2017; Creswell, 1998). The present researcher removed all personal data aspects from the data before storing the research data on a jump drive for seven years then destroying it (Bowen, 2017; Creswell, 1998; Her, 2017).

Summary

Chapter 3 contained a description of this researcher's planned process and procedures to conduct this research. This description included details for the intended purpose, sampling criteria and recruitment procedures, setting, instrument, data collection, analysis as well as the ethical considerations. Next, Chapter 4 will contain details of the data collection process, the results along with the analysis of the findings and results of this qualitative Delphi study.

CHAPTER 4. RESULTS

Introduction

The objectives of Chapter 4 are to present the steps of data collection, identify any enhancements from the planned process in Chapter 3, and document what occurred during data collection. Also, present the results of the data collection, analyze the data collected, and conclude by summarizing the answer to the research question. The data collection results will reflect the interviewer's questions, the interviewer's interpretation, synthesis, and integration of the interviewee's feedback in each round of the Delphi interviewing process, and the confirmation of the emerging consensus among the interviewees including tables and charts as needed. The data analysis will include a presentation of critiques and an assessment of the data as well as figures and tables showing emerging patterns and new narratives.

Modifications and Enhancements to the Data Collection Planned Process

Introduction of incentive for participation in all rounds by the deadline. This modification was due to the little or no response even after sending over 53 LinkedIn invitations and several reminders within the first two weeks of data collection. So, the present researcher decided to add some incentive to encourage faster response. This researcher modified the introduction message to indicate that there will be an incentive for timely responses. The update was to conduct a random draw, using the participants who responded before each deadline of all the rounds, for two prizes of $25 Starbucks gift cards. This researcher made sure not to draw unnecessary attention to the incentive as per IRB guidelines not to present them in bold (Bowen, 2017).

Combining the recruitment, IRB approved informed consent and Round 1. This modification was because most potential participants who responded to the recruitment never came back to respond to the IRB approved informed consent form. Even though the invitation to participate and the introduction indicated that the first step was recruitment, that there would be three rounds of Delphi data collection. The modification included discontinuing use of the initial recruitment questionnaire, and combining the forms so that the potential participants could handle all the steps leading up to Round 1 question in one sitting, which included the following four sections:

1. The introduction to the study along with some definitions
2. The recruitment questionnaire
3. The IRB approved informed consent
4. The Delphi Round 1 question

Furthermore, because some potential participants already responded to the recruitment questions, there were ultimately the following two sets of data and groups of potential participants:

Combo group. The combo-form included the IRB approved informed consent section and Delphi Round 1 section. This group was for respondents to the initial recruitment, meaning they had read the introduction too, and therefore, their next step would be to the combo-form. All respondent would only have access to the Delphi Round 1 section if they chose the "I consent" option; otherwise, they would be sent to the submit section if they chose "I do not consent," (see Appendix C).

Triple group. For potential participants who had not yet started any part of the data collection process. This researcher replaced the link in the invitation with the link of the new triple-form and sent out new invitations to those who had not responded and to new potential participants. The triple-form group had all four sections in one Google Form link including the conditional link to the Delphi Round 1 question if the participant checked “I consent” in the informed consent section. If the potential participant checked “I do not consent” the next section was the submit section as seen in Appendix C.

Clearly stated the use of the collected email addresses. This modification was because after two reminders for potential participants who had responded to the recruitment to respond to the IRB approved informed consent form, one of them informed the present researcher that they rarely check the email address they had provided. So, the present researcher modified the thank-you message after the first visit, to state that they will receive all subsequent communications during the research via the email address they provided. Then, after completing data analysis and personal identifiable information (PII) were no longer needed, the present researcher handled the PII according to IRB guidelines by storing a hard copy of the PII mapping to the codes in a locked cabinet, separate from the rest of the data collected (Bowen, 2017).

Discontinue the use of the LinkedIn group for this research. This modification was because no respondent had joined the LinkedIn group and also because some respondents were from snowballing and did not necessarily have a LinkedIn account. This researcher updated all the forms removing all reference to joining or using the LinkedIn group.

Data Collection Results

Complete Steps of Data Collection

After the modifications, the data collection process, including the modifications were as follows:

- This researcher sent out the Google Form link via LinkedIn direct messages.
- The message also encouraged the potential participants to forward the link to others the thought might be interested.
- When after one week a response was not received, the present researcher sent one reminder every two days, and after reminding thrice, the present researcher concluded that the potential participant was not interested, removed them from the potential participant list and moved on.
- For each response received, the present researcher evaluated them against the inclusion and exclusion criteria to determine if they qualify to participate.
- For the potential participants whose recruitment responses indicated that they qualified to participate, the present researcher sent them an invitation with the link to the IRB approved informed consent via the email they provided in their responses. The invitation also included information on joining a LinkedIn group created for the present research, for communications on deadlines, and participants could ask general questions. The message included reassurance that the present researcher would respond to personal questions individually.

- After one week, only two recruitment responses were received, and after two weeks, there were five responses. However, none of the respondents had joined the LinkedIn group nor signed the informed consent. So, the present researcher created a second form with four sections creating two groups of potential participants (the combo-form group, and the triple-form group). This researcher closed the original recruitment link with a thank-you message, leading the visitors to the new "triple-form." Also, the present researcher updated the introduction message, removing the link to join the LinkedIn group, as well as introducing a random draw for two prizes to those who responded in all rounds within the deadline.
- Potential participants who had responded to the original recruitment form were the combo-form group. The combo-form group received a link to (a) the IRB-approved informed consent, with a conditional "next" button to open the submit and exit button if they did not consent, or (b) the Round 1 question form if they consented to participate. See Appendix B for the researcher-designed recruitment questionnaire and Round 1 question and Appendix C for the transition to exit upon not consenting. This researcher informed the combo-form group of the triple-form if they chose to use that instead.
- The "triple-form group" was everyone else meaning the (a) introduction and recruitment questionnaire, (b) the IRB-approved informed consent, with a conditional "next" button to open the submit and exit button if they did not consent, or (c) the Round 1 question form if they consented to participate. This researcher re-sent the

triple-form to all who had received the original form and not yet responded, to new potential participants, along with a request to forward to others.

- Upon submit, the form displayed a thank-you message, letting the participants know that the link to the next round would be via the email address provided.
- This researcher reached out to a few of participants participant privately, requesting clarification on their responses.
- Also, one participant reached out to this researcher, concerned that he may have misinterpreted the question in Round 1, or responses may not meet this researcher’s expectations because the question was open-ended. This researcher reassured the participant that the question was open-ended because it was about the participants’ experiences, and therefore, there were no researcher expectations. Two other participants reached out to the present researcher apologetically, both because they did not think they qualified due to their lack of experiences with LSAs, and one had already responded to the recruitment and declined the consent.
- Once the present researcher had the desired number of Round 1 responses, the present researcher analyzed the data starting from the recruitment responses. Only those who had consented to participate matched the inclusion criteria and not the exclusion criteria were qualified as participants.
- This researcher combined the five responses from the combo-form group to the 13 responses in the triple-form group in a single spreadsheet for a total of 18 responses,

assigned codes such as Participant 1 (P1), removed email addresses from the rest of the data, and then closed both the combo-form and triple-form links.

- This researcher then analyzed Round 1 responses and prepared questions for Round 2. This researcher invited the 13 participants who had consented and qualified to participate in Round 2 via email, as well as LinkedIn direct messages.
- In Round 2, the present researcher presented the outcome without revealing who said what. The Round 2 form included a list of the six unique strategies from Round 1, and the present researcher asked the participants to choose their most preferred strategy and why it was their most preferred along with the pros and cons (see Appendix D for the researcher-designed Round 2 questionnaire). This researcher gave the participants a one-week period to respond and sent gentle reminders as the deadline approached.
- This researcher received ten responses by the deadline and started data analysis. One week after the deadline, the present researcher received communication from one participant still looking to submit their response. To keep the participation fair, the present researcher did not consider any further submissions and closed out the round.
- This researcher downloaded all the data collected and deleted them from the Google Forms page. Then coded the participants' identity, for example, using "Participant 1", "Participant 2" to protect their integrity, and maintained the key to codes in a separate document, in case the present researcher needed to contact them for clarification at a later point during data analysis.

- After analyzing the data from Round 2 and summarizing the convergence attained along with possible reasoning behind them, the present researcher prepared Round 3 to share Round 2 results with the participants.
- This researcher also used Round 3 to find out from the participants whether they agree with the summaries from all data analysis, and also for the participants to rank their order of preferences of the three prevalent strategies from Round 2.
- This researcher also asked the participants to provide any additional consideration they deemed necessary to understand their responses (see Appendix E for the researcher-designed Round 3 questionnaire).
- All ten participants from Round 2 responded to Round 3 within five days, which was within the seven-day deadline.
- This researcher then pulled down the responses of Round 3 from the google forms, deleted them from google forms, and analyzed the data for additional information to include in the conclusion of this qualitative Delphi study.
- After completing data analysis and PII was no longer needed, the present researcher separated the email addresses and company information of the participants from the rest of the data. The separation was achieved by printing out the email address to Participant '#' mapping, as well as the company name to Organization '#' mapping, to store in a locked cabinet in this researcher's home and deleted the PII from the rest of the data.

Inclusion Criteria

- Should be IT leaders, including but not limited to team leads, IT managers, IT directors, CTOs, CIOs and other decision-makers of SMSEOs.
- Have modernized or are currently modernizing a legacy software application.
- Have led a team of at least five members for at least five years.

Exclusion Criteria:

- No more than two participants from the same company to promote diversity.
- Anyone with a gap experience from the other participants with over 10 years difference (for example if all other participants have between 5 to 20 years of experience and then one person has 31 years of experience, drop the person with 31years).

Recruitment Questions, Rationale, and Results

This researcher requested for email addresses to use for further communication during the research. This researcher requested for the job title, the name of their company, length of time in leadership, size of smallest and largest teams led, to determine if they fit the inclusion criteria leadership role in an SMSEO and are not in the exclusion criteria. The recruitment questionnaire also included questions on the definition of an LSA, what it means to modernize an LSA, number or LSAs modernized or currently being modernized, and length of time used for the modernization and challenges faced, to determine the background of the participant and if the modernization was in an SMSEO all to determine inclusion.

Table 2
Description of Recruitment Respondents

Participant ID	Company	Job Title	Time as IT Lead	Team size	LSA(s) modernized	Meets Inclusion criteria
Participant 1 (P1)	Organization 1	Project Manager	12 years	5-14	12	yes
Participant 2 (P2)	Organization 1	Program Manager	9 years	3-12	7	yes
Participant 3 (P3)	Organization 2	Principal Business Process Technology Analyst	8 years	2-5	5	yes
Participant 4 (P4)	Organization 3	Director of Information Resources	5 years	1-4	2	no
Participant 5 (P5)	Organization 4	Enterprise Architect	5 years	5-9	1	yes
Participant 6 (P6)	Organization 5	Senior Systems Architect	5 years	5-9	5	yes
Participant 7 (P7)	Organization 6	Senior Program Engineer	12 years	4-43	104	yes
Participant 8 (P8)	Organization 7	Systems Analyst	10 yrs. & 9 months.	6-10	8	yes
Participant 9 (P9)	Organization 8	Manager of Support Services	10 years	7-30	2	yes
Participant 10 (P10)	Organization 9	Account Manager	28 years	1-15	10	yes
Participant 11 (P11)	Organization 10	IT Manager	20 years	4-56	5	yes
Participant 12 (P12)	Organization 11	Pastor, Video Editor, MIS	9 years	4-15	2	yes
Participant 13 (P13)	Organization 12	Executive Director, Enterprise Data Management & Engineering Directorate Office of Information & Technology	5 years	10-5000	200	yes
Participant 14 (P14)	Organization 13	Software Architect	10 years	3-7	3	yes
Participant 15 (P15)	Organization 14	Chief Technical Officer	8 years	3-30	3	yes
Participant 16 (P16)	Organization 15	Product Manager	5 years	6-15	5	yes
Participant 17 (P17)	Organization 16	Systems Engineer	3 years	2-3	5	no
Participant 18 (P18)	Organization 17	Network Specialist	1 year	1-3	0	no

Note. The identification of the participants such as email addresses and organization names were removed to protect identities.

As seen in Table 2, 18 individuals responded from both the combo-form and the triple-form. P1 and P2 were from Organization 1, which would have warranted excluding one. However, both did not respond to the consent form along and with P4, therefore excluded. P17 and P18 did not meet the inclusion criteria and therefore excluded from the next rounds.

Table 3
Participants' Definitions of LSAs

Participant ID	Definition of LSA	Theme
P3	Outdated, obsolete, and, not matching current industry trends.	Obsolete, Outdated
P5	Any software or version of software that has been deprecated in favor of newer software, newer versions of software or entirely alternate software.	Obsolete
P6	discrete modules for different business units each with their own storage and application lifecycle	Discrete
P7	It is defined as an old code in a system. In other words, it is code written with older technologies or pattern.	Old code, or technology
P8	An outdated software that an organization uses to continue operations, performs tasks, and generate products, even though other systems has been updated.	Outdated
P9	Outdated program or application that has become end of life and more effective options are available	Outdated
P10	Legacy software applications are outdated or unsupported applications that could be still in use.	Outdated/ unsupported
P11	A legacy software application is a fully implemented and used software application within an organization with no or periodic enhancements to provide additional features and functionality necessary to keep up with changing business needs or regulatory requirements.	Unsupported/ resists update
P12	A software that has gone out of line or that requires an upgrade because it no longer fulfills the purpose(s) of it use.	Obsolete
P13	Applications that are still functional but are reaching/surpassed end of life (support) leading to obsolescence and are now creating technical debt and potentially creating operational or security risk.	Obsolete/ security risk
P14	Outdated or obsolete software	Obsolete/ outdated
P15	Software that cannot be scaled or updated to meet with evolving business needs because the technology used is outdated.	Unscalable/ update-resistant/ outdated
P16	A legacy software application is an application that is used within an organization although the support of the vendor or provider of the application has been stopped.	unsupported

Note. This table contains the responses of only the respondents who qualified as participants after analyzing Table 2. To use later for analyzing how the definition impacts their strategy choice.

Table 4
Participants' Definitions of LSA Modernization

Participant ID	LSA modernization definition
P3	Swapping the legacy system with one designed in response to current industry trends.
P5	Upgrading or replacement of legacy software with newer software, as well migrating the data managed by legacy software to newer and supported platforms.
P6	Implementing a common data model, a reconciliation storage for enterprise-wide reporting, exposing transactional data through APIs, build business processes through reuse of exposed services across the legacy applications
P7	Legacy software application modernization is the upgrading of the obsolete code.
P8	The ability to transfer legacy software capabilities to current, up to date software
P9	Consolidation of software to align with current business needs and infrastructure.
P10	Modernization is modifying or replacing existing legacy applications to bring them up to date and function the original tasks.
P11	Modernizing legacy software involves replacing an existing application or system of applications to leverage new technology to provide the ability to meet current or future business needs.
P12	This is the upgrading of existing software to meet up with current operational and functional demands.
P13	Legacy application modernization has as its focus creating business value by re-factoring, consolidation of capabilities, repurposing and aligning applications to meet business demand. Cloud computing is a great example of IT alignment and re-factoring to achieve business goals with greater celerity and often reduce costs. Portfolio rationalization, and application rationalization are central to any IT modernization.
P14	Move apps to newer or better technologies
P15	Legacy software modernization is the rewriting software in such a way that it is adaptive which means that it can be updated and scaled to meet with changing business requirements by using latest technologies.
P16	A legacy software application modernization is the process of replacing or renewing part or the totality of the legacy application by newer and more maintainable software. Depending on the context in which the software is used, one would either choose to replace or renew parts or the totality of the application. Within the projects I have worked in, one would generally renew parts of the software as long as it would work because this could be handled within a maintenance contract the software provider had with the client. Replacing the software application was almost always related to a bigger project because the development team, the maintenance team, and the users had to be trained to use the new technology.

Note. The respondents also provided their understanding of what LSA modernization means to help set the pace for and expand on the reason for their choice of strategy.

The data collection plan had been to start with 20 participants so that if some participants fell off during the different rounds, there would hopefully still be ideally about 12 left. However, since the recruitment had taken about a month already, the present researcher closed the recruitment phase with the 18 total responses. Another recruitment question was the participants' definitions of an LSA and LSA modernization (as seen in Table 3). One more recruitment question was the challenges that they have faced in LSA modernization so far. The response to this question would help analyze the participants' approaches to selecting a modernization strategy.

Round 1 Question, Rationale, and Results

P1 and P2 work in the same organization, so only one of them could move forward. However, three participants (P1, P2, and P4) from the combo-form group never signed the informed consent and therefore automatically dropped from Round 1. As for the triple-form group, after reading the introduction, Participant 18 chose not to participate due to a lack of modernization experience and informed the present researcher privately. This researcher dropped P17 due to the missing inclusion criteria in the five years leadership experience, and also because though they listed that they had modernized five LSAs, their response to the Round 1 question was that they had not thought about how to modernize yet.

The researcher designed Round 1 question was: What strategies have you used, or would you use to modernize the application? Please Identify the strategy name if it has one, and describe the steps used for the modernization, along with why you think it is the best approach for you. The participants had to enter a minimum of 1500 characters. The question was to ensure that the participants would outline their experience in modernization.

Round 2 Question, Rationale, and Results

As seen in Round 1 data collected above, of the 13 participants considered, there were 15 total strategies. For Round 2, the present researcher compiled the 15 strategies into six unique strategies and presented them as a list for the participants from which to choose. This researcher also asked the participants to provide a reason for their choice along with the pros and cons of their choice. The rationale of the question was to observe the participants' choice and the reason for their choices after reading the strategy choices of other IT leads. Of the 13 qualified participants in Round 2, ten responded to the Delphi Round 2 questions and selected their preferred strategy as well as provided the reason for their choice and the pros and cons.

Round 3 Question, Rationale, and Results

As seen in Round 2 data collected above, there were three prevalent strategies, namely Strategy 2 at 80%, Strategy 4 at 10%, and Strategy 6 at 10%. For Round 3, the present researcher compiled the pros and cons of the three prevalent strategies. While mapping the pros and cons back to what the participants had provided as their challenges in modernizing the present researcher created some summaries as to why the participants may have made their choice in strategy. This researcher also asked the participants to rank their order of preference of the three prevalent strategies and offer any additional information they deemed necessary to understand their choices better. Ninety percent of the participants agreed with both the summaries and that Strategy 2 is the common LSA modernization strategy, while 10% decided that while they agreed with the summary, they preferred Strategy 6 as the common strategy, and the ranking also aligned with the choices.

Data Analysis and Results

Round 1 Data

There was a total of 15 strategies in response to the Round 1 question because two participants gave more than one strategy. The summarized strategy findings for further analysis were as follows:

1. **Integrate with DevOps:** Ensure that regardless of the chosen solution for the specific case, the steps to implement should include incorporation with modern DevOps automated (push-button) mechanism, irrespective of the application size. This integration promotes a continuous integration process. Example Content Integration/Content Delivery (CI/CD) processes such as the Microsoft Azure DevOps and the AWS Code (both cloud-based).
2. **Unchain application from infrastructure, then abstracted and detached.** Detach from any dependencies in the main infrastructure. Abstract the functions/tasks of software into independent components that can run anywhere (micro-applications), then integrate the components in a new infrastructure amalgamation, thereby modernizing both the application and the infrastructure concurrently.
3. **Architecture driven modernization**. Replace with new development and use shared resources in the cloud.

4. **Sherwood Applied Business Security Architecture (SABSA):** Base every decision regarding the modernization on the business security requirements. Sherwood Applied Business Security Architecture To facilitate business. No license required steps: analyze business requirements, Use SABSA top-down model to create a traceability chain (planning, design, implement, ongoing manage & measure (preserve the business), using SABSA Matrix and business attribute profile.
    - Extract extractable pieces to the cloud as SaaS
    - Prioritize components of the application by criticality
    - List and prioritize LSA exploitable vulnerabilities
    - Conduct a risk assessment to ensure value is greater than the cost
    - Define data loss prevention safeguards
    - Conduct a benefit analysis to determine the best return on investment (ROI)
5. **Incremental modernization**. Starting with the least critical aspect of the application Using Agile SCRUM project management. This action will make use of user data as they use the incrementally modernized pieces to help build the documentation.
6. **The Greenfield approach**. Given available freed up capital along with the IT and business teams working hand in hand, the Greenfield approach is best because there are no constraints from prior work.
7. **Rearchitecting or refactoring incorporated with testing**. This strategy involves rehosting, refactoring, rearchitecting, rebuilding, and replacing. The strategy must factor in security loopholes.

8. **Project management approach.** The steps include management support, a thorough understanding of the scope and size of the legacy application for example number of users, number of interfaces with other applications, how critical is the application, how complex is the application.

    - Only proceed if sufficient resources such as money and staff are available to complete a project based on estimation
    - Document accurate and complete business rules of the application, then create a project plan
    - Include level of effort (LOE) estimates
    - Prepare a test environment to mirror production in infrastructure and setting
    - Update and involve the users and leadership in the project plan
    - Prepare training material
    - COTS application option
    - Choose a project management technique preferably Agile

9. **Problem-solving oriented method.**

    - Analysis of need and requirement for modernization
    - Design the structure and procedure of the actual modernization (upgrade or replacement)
    - Backup existing data
    - Execute and install modernization

- Test the application perform troubleshooting quality assurance and regression testing
- Train staff on modernization made

10. **Replace with new development**. Get a good understanding of the legacy application's architecture including technology stack, interfacing applications, and organize a demo of the application, and the use cases the prepare some sort of a document especially if documentation is unavailable or incomplete.
    - Prepare a project plan for the entire application, choose most used technology and third-party integration where possible, recruit qualified staff,
    - Minimize disruption of critical applications relying on the legacy system by starting with the least critical components.
    - Use API driven development by creating incremental chunks as APIs
    - Identify and back up data
    - Use test-driven Agile development and automate testing where possible
    - Also, include weekly demos of working pieces as training and energy booster of the staff.
11. **Replace with microservices**. Identify functional units to convert to a microservice, assign each functional unit to an Agile team, each team will create user stories using Gherkin (Given/when/then) for easy test cases, prepare the deployment environment and process using containers for easy deployment, use continuous integration, set up monitoring processes and a team to handle issues that come up.

12. **Multi-Phase Replacement**.
    - Determine and define the data structures and business logic from the legacy application
    - Determine what to change and what to maintain
    - Perform Gap analysis of what is covered in the legacy application and what is potentially not covered in the replacement.
    - Complete test-driven implementation of the replacement application, both data, and logic wise, using Agile methods and stories.
    - Run the new application parallel to the old until the modernization is complete.
13. **Continuous integration using Agile**. Use DevOps tools to automate monitoring and testing. E.g., SecDevOps within CI/CD pipeline, detect security flaws resulting in broken builds, decide between DevSecOps, SecDevOps, and SecOpsDev addresses design review, input validation, isolation of untrusted inputs, performing compliance requirements, performing security configurations, performing security policies, security requirements analysis, performing manual security test, risk analysis, and threat modeling.
14. **Design, Transition, Operations, and continuous improvement**. Using Agile development
15. **Use a COTS solution**. Ensure thorough communication and training on the modernization reason and procedures to avoid employee turnover. Have a business process review one year after implementation.

Table 5
Constant Comparison Analysis of Round 1 Data

Response Chunks	Code for each chunk
incorporation with modern DevOps automated (push-button)	DevOps tools (migration)
Content Integration/Content Delivery (CI/CD)	CI/CD (Migration)
AzureDevOps and the AWS Code (both cloud-based)	Cloud (migration)
Use DevOps tools to automate monitoring, and testing	DevOps tools (migration)
SecDevOps within CI/CD pipeline	CI/CD (Migration)
Security requirements analysis	Analysis (problem-solving)
Cloud computing is a great example of IT alignment and refactoring to achieve business goals with greater celerity and often reduce costs.	Cloud (migration)
SecDevOps (continuous integration/continuous deployment pipelines)	CI/CD (Migration)
Scripts-Amazon, Azure, IBM, and many other Cloud service providers	Cloud (migration)
Improves security SecDevOps an excellent way to build security into the application by addressing code coverage and known vulnerabilities	code coverage (migration)
DevSecOps includes process changes, such as establishing strong feedback loops within the organization, performing code audits on a regularly scheduled basis to ensure quality, standards compliance,	DevOps tools (migration)
Integrate with DevOps	DevOps tools (migration)
Upgrading legacy software applications is inherently quicker with a point-and-click, push-button approach	DevOps tools (migration)
Migrate the legacy application to new infrastructure amalgamations, without altering a single line of code in the legacy software applications that may undergo upgrade at any time	new infrastructure (migration)
Both build and release pipelines are configured for CI/CD (Content Integration/Content Delivery) processes	CI/CD (Migration)
Continuous integrations process has an artifact build process that can convert legacy software applications on a push of a button and have the application released as an upgraded version	CI/CD (Migration)
Sherwood Applied Business Security Architecture	SABSA
Architecture Driven Modernization	SABSA
Unchain application from infrastructure, then abstracted and detached	abstract (replace)
Define the data structures (schema) as well as business logic in the source legacy application	extract (replace)
Define what critical data and logic from this source application is actually important to the business today	documenting (replace)
Map the data and business logic from the legacy app to the new app	documenting (replace)
Gap analysis of missing information in the new system that is covered in the old system	documenting (replace)

Response Chunks	Code for each chunk
Test data migration pipelines are created to load data into the new system from the legacy app	Data handling (replace)
Project Management is employed to start testing plans to start testing all these pipelines and new logic.	testing (replace)
Application features tested in the new application	testing (replace)
New system starts running in parallel with the legacy app	parallel (replace)
Data sources and the network and security formations, as well as application data, can all be abstracted	abstract (replace)
Upgrading the infrastructure at a scheduled time as designated project without interring with the operation of the legacy applications	parallel (replace)
API driven development by creating incremental chunks as APIs	micro-services (replace)
Agile	project management (replace)
Greenfield approach when capital is freed up for this transformation and in most cases this transformation requires both the business and IT to work closely together	greenfield (replace)
How does the new software integrate with other applications within your environment	change management (replace)
How do training and skill-sets fit with the new application	training (replace)
an iteration process of moving off legacy software	iteration (replace)
How it will affect staff or even consumers	change management (replace)
Developed in-house, using Agile methodology	replace (new)
Making enhancements to COTS application	enhancements (replace COTS)
Business process reviews, one year after	review (replace COTS)
Replace with new development	replace (new)
Replace with micro-services	micro-services (replace)
Create user stories using Gherkin (Given/when/then) for easy test cases, prepare the deployment environment and process using containers	replace (new)
When the upgrade is possible with the old system, an analysis is done to find out what it will take in terms of design (redesign), time, work disruption and financial cost.	Analysis (problem-solving)
A deliberation on the cost-benefit and what may be the long-term cost of not upgrading	Analysis (problem-solving)
Data migration from one system to a newer version	data migration (migration)
Rehosting, refactoring, rearchitecting, rebuilding and replacing as optional paths when making the journey to the cloud as well as factor in security loopholes	refactoring
Data backup	problem solving

Note. All 15 strategy entries were searched for the data chunk to determine the different use-cases and reoccurring themes. COTS applications in the table stand for commercial-off-the-shelf applications.

The participants had chances to clarify some of the responses as needed, by updating their original entry in the Google Forms, and when the response was quick enough like typos, the participants let the present researcher know. This researcher analyzed the 15 responses by applying a combination of constant comparison analysis (CCompA), and classical content analysis (CContA), and using the existing modernization strategies in Table 1 as a guideline. This researcher then coded the recurring themes into narrower codes in search of convergence. As seen in Table 6, six recurring themes emerged.

Table 6
Recurring Themes Mapped to Data Analysis Codes

Codes for data chunks	Recurring themes
CI/CD (Migration)	migration
DevOps tools (migration)	migration
Cloud (migration)	migration
code coverage (migration)	migration
new infrastructure	migration
Architecture Driven Modernization	SABSA
change management (replace)	replace (new or COTS)
project management (replace)	replace (new or COTS)
training	replace (new or COTS)
greenfield (replace)	replace (new)
micro-services (replace)	replace (new)
parallel	replace (new)
abstract	replace (new)
testing	replace (new)
extract	replace (new)
documenting	replace (new
data handling	replace (new)
iteration (replace)	replace (new)
enhancements (replace COTS)	replace (COTS)
review	replace (COTS)
Analysis (problem-solving)	problem solving
data migration (migration)	migration
refactoring	refactoring

Note. Showing the CCompA findings. CI/CD stands for content integration/content delivery

This researcher also used classical content analysis (CContA) to determine the popularity of each recurring theme, as seen in Figure 2.

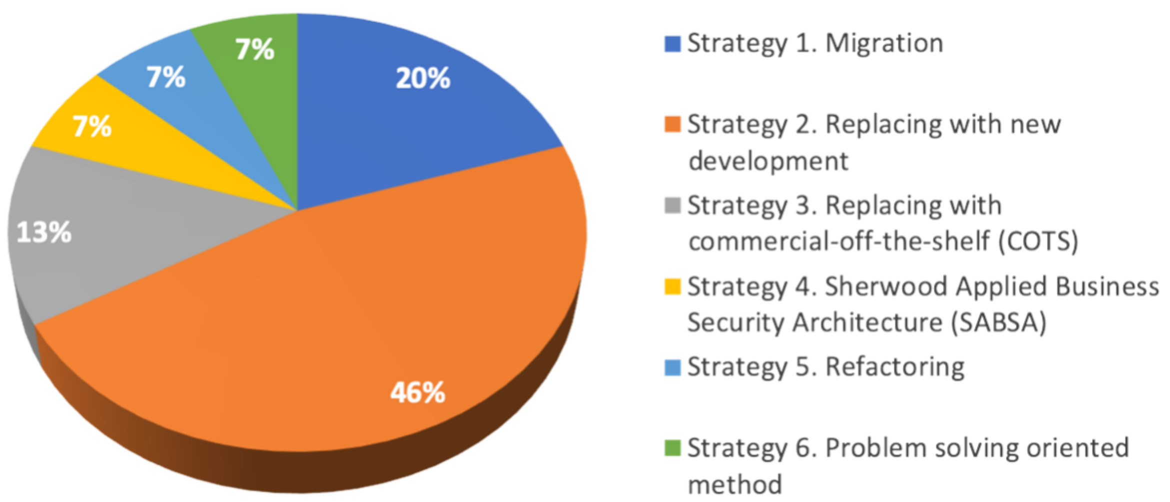


Figure 2. Showing the result of the classical content analysis (CContA). A total of 15 opinions out of the 13 responses, and there were three that converged into Strategy 1, seven into Strategy 2, two into Strategy 3, then one each into Strategies 4, 5 & 6.

This researcher formally composed the data analysis results into six unique strategies to use in the Delphi Round 2 question.

Strategy 1. Migration. Continuous integration using cloud-based DevOps tools such as Content Integration/Content Delivery (CI/CD) processes such as the Microsoft Azure DevOps, DevSecOps, SevDevOps, DevOpsSec, and the AWS Code. The utilization of the DevOps tools will help to address:

1. Design review
2. Input validation

3. Isolation of untrusted inputs
4. Performing compliance requirements
5. Performing security configurations
6. Performing security policies
7. Security requirements analysis
8. Performing manual security test
9. Risk analysis
10. Threat modeling

Strategy 2. Replacing with new development. Abstract the functions or tasks of the legacy software into independent components that can run anywhere and separate them from the infrastructure, then create microservices for each component. The components should be prioritized and then replace them, starting with the least critical or concurrently by assigning the various components to different teams. Use Agile processes and continuous integration to achieve this goal. For each implemented piece, set up monitoring processes and assign to a team for monitoring them. It is crucial to understand the LSA's architecture including technology stack, interfacing applications, and organize a demo of the application, and the use cases the prepare some sort of a document especially if documentation is unavailable or incomplete. This approach would be the Greenfield approach where, besides the business logic, the new application does not adhere to the constraints of a pre-existing system. Some steps in this strategy include:

1. Analyze the legacy application for the number of users, the number of interfaces with other applications, the criticality, and complexity of the application.
2. Document accurate and complete business rules of the application.
3. Determine and define the data structures in the legacy application.
4. Determine what to change and what needs to maintain.
5. Perform Gap analysis of the LSA coverage and what the replacement potentially does not cover, given the new technology and the business logic changes.
6. Prepare a project plan for the entire application, choose most used technology and third-party integration where possible, recruit qualified staff.
7. Minimize disruption of critical applications relying on the legacy system by implementing the least critical components first.
8. Use API driven development by creating incremental chunks as APIs.
9. Identify and back up data from each LSA component.
10. Use test-driven Agile development and automate testing, where possible, ensure that the testing environment mirrors the production environment.
11. Include weekly demo of working pieces as training and energy booster of the staff.
12. Run the new application parallel to the old until the modernization is complete.

Strategy 3. Replacing with commercial-off-the-shelf (COTS). Ensure there is adequate communication modernization reason and training on the procedures, to avoid employee turnover. Have a business process review one year after implementation.

Strategy 4. Sherwood Applied Business Security Architecture (SABSA). Base every decision regarding the modernization on the business security requirements. SABSA to facilitate business and analyze business requirements. Use SABSA top-down model to create a traceability chain (plan, design, implement, manage & measure), preserve the business, as well as apply SABSA Matrix and business attribute profile without requiring a license. Required steps are:

1. Extract extractable pieces to the cloud as SaaS
2. Prioritize components of the application by criticality
3. List and prioritize exploitable vulnerabilities
4. Conduct a risk assessment to ensure value is higher than the cost
5. Define data loss prevention safeguards,
6. Conduct a benefit analysis to determine the best return on investment (ROI)

Strategy 5. Refactoring. Includes rehosting, rearchitecting, rebuilding, and replacing, as well as factoring in security loopholes.

Strategy 6. Problem-solving oriented method.

1. Analysis of need and requirement for modernization
2. Design the structure and procedure of the actual modernization (upgrade or replacement)
3. Backup existing data
4. Execute and install modernization
5. Test the application perform troubleshooting quality assurance and regression testing
6. Train staff on modernization made

Round 2 Data

In the Delphi Round 2, the present researcher sent the invitation to participate via email and LinkedIn direct message to the 13 qualified participants. The participants were provided the feedback on Round 1 and asked to review the six strategies that resulted from Round 1, then select the one they would use over all the other and provide a reason for the choice, as well as the pros and cons of their choice. This researcher gave the participants a one-week period to respond. This researcher sent reminders as the deadline drew near and received a total of 10 replies within the one-week timeframe.

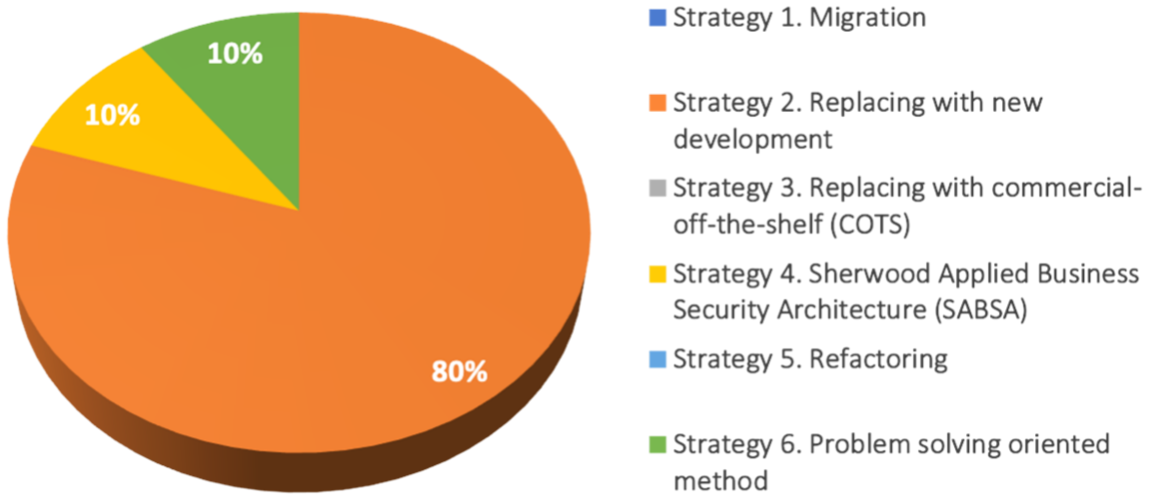


Figure 3. One participant selected Strategy 4, one participant Strategy 6, and eight participants Strategy 2 for a 10% to 10% to 80% strategy preference.

As part of the responses, the participants also provided the reason for their choice along with the pros and cons of their preferred modernization strategy. This researcher found that nine of the pros of the chosen strategies aligned with the comprehensiveness of the strategy, three toward having a fresh perspective, manageable, Traceable, customizable, and cost-effective. There were two occurrences each for risk management, adaptable, flexibility, and compatibility, and the rest occurrence each.

Table 7
Pros of Round 2 Modernization Strategies

Strategy	Pros	Keywords
Strategy 2	Comprehensive enough to encompass the pros of all the other stated strategies while minimizing the cons.	comprehensive
	Brings in the aspect of priority by implementing the least critical first (limiting the impact of risks as the replacement process moves on)	Risk management
	Original choice in Round 1 and successfully delivered to production	successful
	It will give a fresh perspective on how the final product will be developed keeping in mind known challenges and future enhancements	fresh perspective
	It allows the implementer to breakdown the process into manageable pieces on which decisions can be made. By breaking down the process into various components, the cost can easily be managed	manageable, traceable, Cost-effective
	Use of open-source technology which reduces cost on licensing	manageable technology and Cost-effective
	If the application has a customer facing user interface (UI), it can be made responsive to have a better user experience across multiple devices that will range from phones, tablets, laptops, and desktops. Rapid and frequent changes can easily be adapted into a new development in response to customer demand as the market evolves	Adaptable to the user and customer needs
	Hundred percent owned by the company	autonomy
	An in-depth approach to modernization	Comprehensive
	The new system will benefit from documentation the data structures and business rules	Fresh perspective
	Combining test-driven development with Agile development will provide a robust process	comprehensive (testing)
	When possible, running the new system concurrently with the legacy system provides an incredibly powerful means to validate the new system is functioning as expected (also provided in Round 1)	Comprehensive (parallel running)
	Simultaneously a very structured methodology as well as a flexible approach that covers all of the major areas in application development (ensure that business rules factor security and post-implementation review)	Customizable
	Replacing LSA is a practical strategy and gives the most value to the business compared to the other strategies	Valuable

Strategy	Pros	Keywords
	Begins with clearly mapping out the as-is functionality from a business, data, application and technology perspectives	comprehensive
	Even though the strategy does not specify security considerations, it can be defined in the business logic.	flexibility, customizable
	Most importantly, the documentation of the functionality of the platform would be scattered across different functional domains due to changes from supplier upgrades to In-house maintenance updates	autonomy (company owned)
	The initial as-is analysis gives a holistic view of the legacy platform and permits stakeholders build a weighted matrix of its functional components	comprehensive
	The new development comes with new technology. The business would have changed from the legacy platform once implemented, the data structures may have changed, the technological stack has moved on, and the interfacing applications have changed.	Fresh perspective
	Newer hardware products can no longer function with old legacy software applications, that will require several tweaks and changing compatibility methods within the application just to make sure that it can run on the new hardware product	Compatible with new hardware
	If budget is not an issue, I will always opt for new implementation as an IT lead, easier to motivate team to participate, more familiarity with business processes, and room for process improvement.	Fresh perspective, motivation
Strategy 4	Deliver consistent, comprehensive and business aligned architecture every time	Comprehensive and consistence
	It ensures that everything I do is traceable back to a business requirement, whilst ensuring that the architecture is designed to manage security risks across	Risk management, traceable
	It is an open standard, comprising several frameworks, models, methods, and processes, free for use by all, with no licensing required for end-user organizations that make use of the standard in developing and implementing architectures and solutions	Cost-effective (no licensing)
	We identified software that could migrate to Software-as-a-Service (SaaS) to the cloud	Migratable
	Abstract to manageable components	manageable, comprehensive
Strategy 6	Business alignment	Traceable
	Problem-solving method allows an organization to define solutions on the process to move in a direction to modernize. Developing a structured process on what next steps during each stage.	Customizable, comprehensive, adaptable, flexibility, compatible

Note. The participants provided their reason for choosing a strategy along with the pros and cons. Presentation of the pros along with the coding for CCompA data analysis.

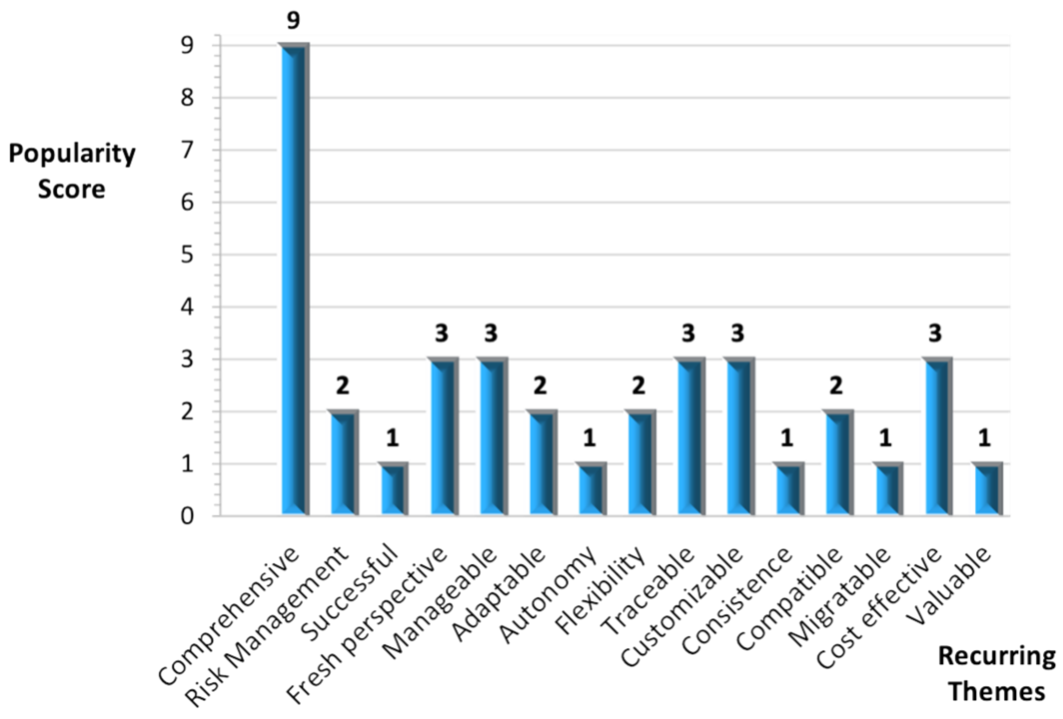


Figure 4. A CContA representation of the pros of the strategies chosen in Round 2, showing comprehensive with the highest appearance of nine. There was 37 total and 15 unique pros, the numbers represent the number of times each pro was mentioned by the participants.

The participants also provided cons to their choice in strategy. However, they only provided a few cons relative to the pros. There were 21 pros to four cons for Strategy 2 solely because Strategy listed “API driven development” in one of the steps, the feedback was that specifying “API driven development” made the strategy too specific. Some of the cons included possible high upfront cost, the need to involve the company executives in the modernization process because the whole migration may be an expensive venture. As for Strategy 6, the was no cons provided and only one con for Strategy 4 namely that the modernization process is too specific. So, there was an immediate convergence with Strategy 2 being the most prevalent preference along with the most pros and cons.

Round 3 Data

In the Delphi Round 3, the present researcher sent the invitation to the ten participants who responded to Round 2 with a one-week deadline. All ten participants responded with five days. The purpose for the third round was to present the findings from Round 2, and the conclusion drawn from all data collected in the study to the participants, then find out if they concur with them or not. Also, have the participants rank the three prevalent strategies from Round 2 in their order of preference and also provide any additional information they deemed necessary to help the present researcher understand their opinions. This researcher also provided the six strategies that resulted from Round 1 to serve as a reminder. For the agreement question, the present researcher provided four options for the participants to choose from to ensure that all bases were covered, and the present researcher made no unintended assumptions. See Figure 5 for the results of the agreement question.

The options were, I agree with the summary and that Strategy 2 is the common strategy, I agree with the summary but do not agree that Strategy 2 is the common strategy, I do not agree with the summary but agree that Strategy 2 is the common strategy, and I do not agree with the summary nor that Strategy 2 is the common strategy.

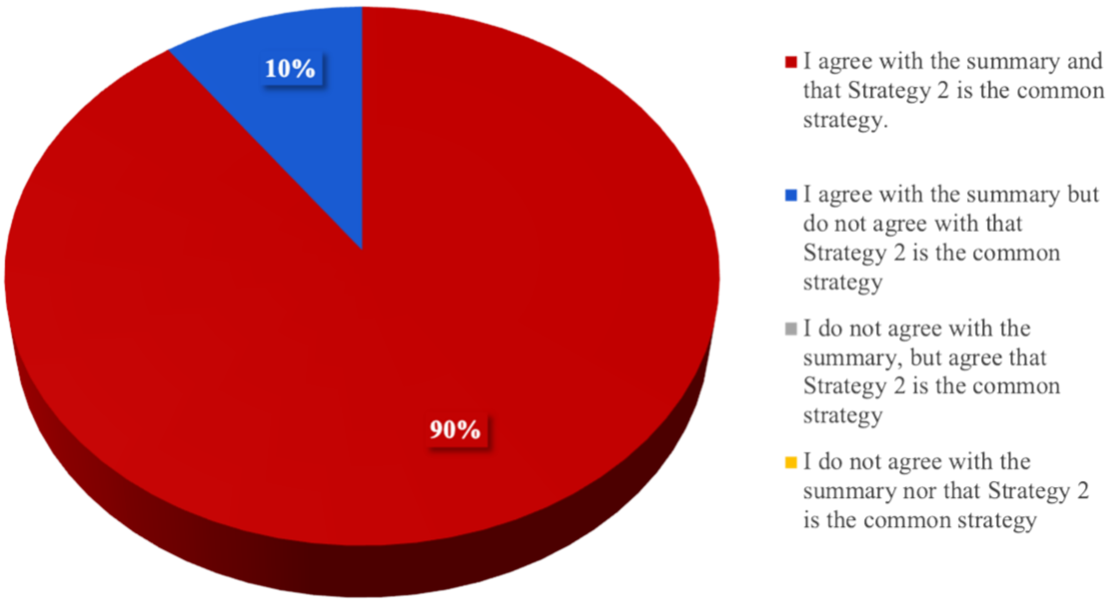


Figure 5. Delphi Round 3 responses showing a 90% agreement with this researcher's summary of the data from the rounds and the emerged common Strategy 2, as well as a 10% agreement with the summary but not with the common Strategy 6.

The agreement in Figure 5 also aligned with the data from the two ranking questions that the present researcher posed to understand the preferred choice of the present researcher in case there was a disagreement with the summary. The ranking of the participant who did not agree that Strategy 2 was the common strategy showed the order Strategy 6, Strategy 2 and then Strategy 4 in both ranking questions and aligned with the participants' choices in strategy in Round 2 but differed from the Round 1 experience pointing to Strategy 2 and Strategy 3. Another participant even after agreeing with both the summary and the common strategy expanded that Strategy 3 would work equally well. See Figure 6 and Figure 7 for the ranking results.

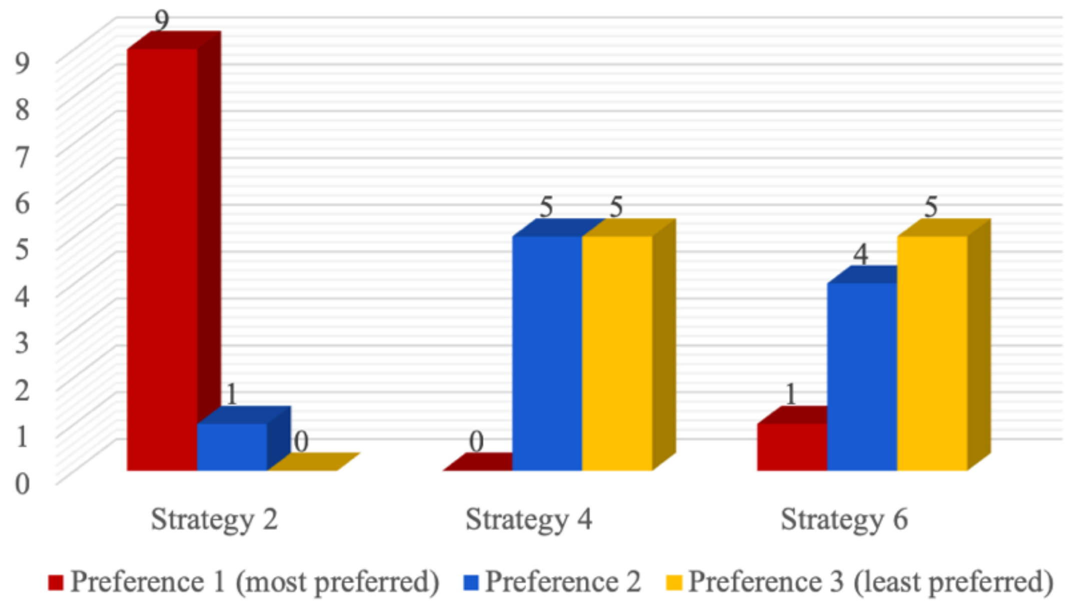


Figure 6. The ranking results showing Strategy 2 as the most preferred strategy with nine votes and one vote for Strategy 6 as most preferred, four for preference 2 and five for preference 3. Strategy 4 came in with five preference 2 and five preference 3.

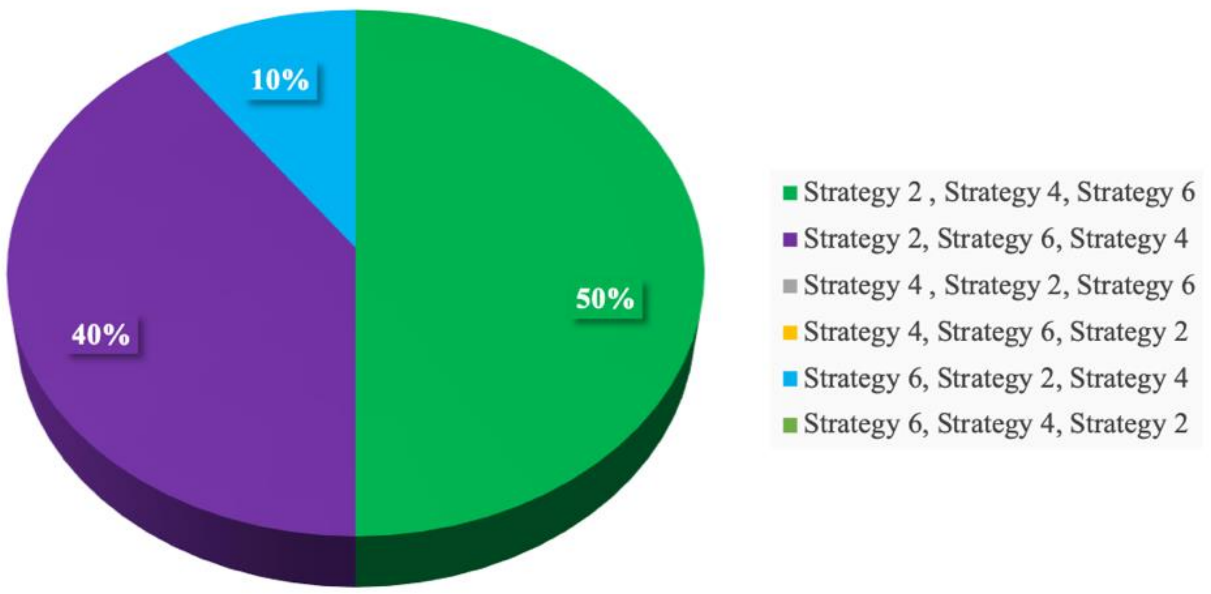


Figure 7. Showing the ranking order of the strategies which align with the ranking in Figure 6.

Summary

Chapter 4 presented the findings from the various Delphi rounds, revealing a common LSA modernization strategy as Strategy 2 (Replacement with new development). This researcher conducted a three-round Delphi study with 13 responses in Round 1, and ten responses each for both Round 2 and Round 3, all from IT leads of various job titles. A critical observation was that the participants who proposed Strategy 2 in Round 1 had job titles such as Enterprise Architect, Senior Program Engineer, Manager of Support Services, IT Manager, Software Architect, Chief Technical Officer (CTO), and Product Manager. As for the consensus, in Round 2, eight participants (80%) chose Strategy 2 as their preferred strategy over the other five strategies proposed. One participant (10%) preferred Strategy 4 (SABSA), for similar reasons that the 80% chose Strategy 2 such as Comprehensive, risk management, alignment with business needs, no licensing needed, and consistency. The final 10% (one participant) chose Strategy 6 (problem-solving strategy) for its customizability which can also be observed to align with one of the pros of Strategy 2. However, in Round 3, there was more convergence when 90% agreed with this researcher's summary of all the data and Strategy 2 as the common LSA modernization strategy, while 10% agreed with the summary but not with the strategy. Given the observation of the job titles of the participants who proposed Strategy 2 in Round 1, another conclusion was the tendency for those in higher leadership roles to propose a strategy that the other participants quickly changed their opinions in favor of and accepted it as the common LSA modernization strategy for most SMSEOs.

CHAPTER 5. DISCUSSION, IMPLICATIONS, RECOMMENDATIONS

Introduction

Chapter 5 covers the evaluation of the research questions, factoring in the research purpose, theoretical framework, and the data collected, to determine the purpose fulfillment of this research. Chapter 5 also presents the contribution of the results to the business technical problem as well as recommendations for further research and a conclusion of the study.

Evaluation of Research Questions

The overarching research question was: What common modernization strategy can IT leaders in SMSEOs leverage to modernize their indispensable LSAs as technology changes? To answer to this overall research question, the present researcher used a two-round Delphi technique. This researcher collected and analyzed data to determine convergence and create the follow-up questions in the subsequent rounds:

SQ 1. What business or technical imperatives drive the modernization of LSAs in SMSEOs?

SQ 2. What challenges impact the modernization process?

SQ 3. What factors contribute to the success of the modernization?

The thought process behind the sub-questions was that first; the LSA should be a problem that the SMSEO is motivated to resolve, hence the question on the imperatives for modernization. Then to further emphasize on the business problem, the present researcher wanted to find out what challenges the IT leads in SMSEOs currently face in modernizing LSAs.

Finally, the present researcher wanted to determine what factors attribute to a successful LSA modernization.

Table 8
The Various Challenges Faced in LSA Modernization and Links to Strategy Pros and Cons

Participant ID	What challenges have you faced with legacy application modernization to date?	Align to Strategy pros	Strategy cons
P3	Resistance to change by potential users; Data extraction, translation, and loading challenges; Running parallel transactions during transition from legacy to new system.	Comprehensive/fresh perspective/Adaptable/Flexibility/Compatible	
P5	Challenging migration paths, unsupported hardware, difficulty to find drivers (APIs) to port data etc...	Comprehensive/flexible	Too specific
P6	resistant IT departments, difficulties with retrieving tacit knowledge, enterprise-wide data model implementation, inefficient executive sponsorship	Traceable/Autonomy	Executive involvement
P7	Compatibility and re-Integration	Comprehensive/Adaptable/ Compatible	
P8	Rapid evolution of software, but slow tech refresh of hardware systems. Testing and production of software, compatibility and security issues moving resources to the cloud. The challenges faced were compatibility issues, training end users, defined or redefined roadmap and risk management plan.	Customizable/Flexible/Fresh perspective	
P9	The biggest challenge is cost moving to a more current version or another application that supports our environment	Open source/customizable	Expensive
P10	Data transfer and training.	Comprehensive / customizable/ consistency	
P11	I witnessed many challenges in all of the projects I participated in regarding replacing legacy applications with new applications. Some of the challenges are: (1) There either wasn't any documentation, or the existing documentation was incomplete and outdated as to how the existing application supported the current business processes. This is critically important whether the new application is built from scratch in house or a commercial of the shelf (COTS) application is bought to replace the legacy application. Significant delays occurred because the project team had to document the business processes. (2) The initial plan was to replace the entire legacy system in one big implementation. After months of developing the plan, it became obvious a "big bang" approach was too aggressive and risky.	Comprehensive /Adaptable/ flexible/	Executive involvement

Participant ID	What challenges have you faced with legacy application modernization to date?	Align to Strategy pros	Strategy cons
	(3) Not enough involvement with the users of the applications on the project team resulting in missing critical functionality that should have been caught much earlier during the analysis and design phase of the project. (4) Lack of attention and resources allocated to training material and actual training of the users of the new application. (5) The inability to stress test the new application with production-like hardware to ensure the new system would meet response time and capacity requirements. (6) Senior management providing an unrealistic time frame and allocating too few qualified resources on the project team to successfully replace the legacy application.		
P12	Usually the upgrade requires time, financial cost, delay in the use of the system, possible data loss or conflicts and bugs, resistance from staff.	Comprehensive/fresh perspective	Expensive
P13	Users not understanding what to use, so I have to make sure they are using the right source to download legacy software	Adaptable/comprehensive	
P14	Db issues and broken third-party integrations	Open source/flexibility	Too specific
P15	Access to the engineers who originally came up with the design and developed the legacy application making it difficult sometimes to have questions answered about certain implementation in the code.	Comprehensive/fresh perspective/ Greenfield	
P16	The lack of documentation. The developers of the software were not in the project anymore. Insufficient documentation from the providers. Most of the software knowledge was in the heads of the developers.	Comprehensive/fresh perspective/ Greenfield	

Note. This table ties in the challenges that the IT leads have faced with modernizing LSA to the pros and cons of the prevalent strategies from this qualitative Delphi study has shed light on why the participants chose the strategies that they did.

For IT leads that have modernized LSAs before they may have one or more strategies they have applied in the past, and for those currently modernizing, they could shed light on the strategy they are currently using and its impact. In the recruitment phases, the present researcher asked the participants to point out some challenges they have faced during LSA modernization. The responses to that question helped the present researcher determine some of the shared imperative drives for modernizing LSAs to change resistance, incompatible to new technology, not migratable, expensive to run and maintain LSA as-is, inflexibility, unmanageable LSA, incomprehensive due to lack of training on dated technology or lack of documentation on LSA, or unpredictability of LSA. This determination answered SQ 1. The answer for SQ 2 was more straightforward from the responses, as seen in Table 8. To respond to SQ 3, the present researcher aligned the challenges to pros from the strategies collected in Round 2, as seen in Table 8.

This researcher was able to determine that the participants' proposed strategies in Round 1, and selected strategies in Round 2 depending on the challenges they faced and were trying to overcome from their previous or current LSA modernization experience. Table 8 indicated that the big challenge with modernizing legacy application in the past was the lack of documentation or the complex and incomprehensive nature of the LSA. Figure 4 showed that comprehensive was the most prevalent opinion on the list of cons of the three prevalent strategies from Round 2 data, and P3, P10, P11, P13, P15, and P16 all had challenges that align with the complex or incomprehensive nature of LSAs and they all chose Strategy 2 as their preferred strategy associating their reason to the comprehensive nature of the strategy. P6 had challenges with traceability of the business requirements from the LSAs and the autonomy of the company over the LSA, and these aspects are all covered in Strategy 2, which was also P6's preferred strategy. P14 had modernization challenges that tied in with the inflexibility of the LSA, and P14's preference of Strategy 2 ties in with the solution to the modernization challenged faced in the past.

These observations indicated that the most prevalent pros of a strategy, which is the detailed and comprehensive nature, aligned to the challenges the participants have faced in the past, which also explained why Strategy 2 was the most prevalent choice.

Fulfillment of Research Purpose

The purpose of this qualitative Delphi research study was to identify a common strategy for modernizing indispensable LSAs (El-Gazzar et al., 2016). Many IT managers in SMSEO have no common strategy for modernizing their indispensable business-critical LSAs as technology changes at a fast pace, causing the modernization process to be inefficient and costly (Crotty & Horrocks, 2017; Morton et al., 2015; Rai et al., 2015). Identifying a common modernization strategy would help the SMSEOs to be able to plan for and budget for the modernization of their LSAs to boost revenue generation from the LSAs while continuing to add value to customer experience (Letier et al., 2014). As seen in Chapter 4, the participants converged at the 90% rate to Strategy 2 as the common LSA modernization strategy. Strategy 2 summarized as LSA replacement with new development.

Replacing LSAs with New Development as a Common LSA Modernization Strategy

Abstract the functions or tasks of the legacy software into independent components that can run anywhere and separate them from the infrastructure, then create microservices for each component. The components should be prioritized and then replace them, starting with the least critical or concurrently by assigning the various components to different teams. Use Agile processes and continuous integration to achieve this goal. For each implemented piece, set up monitoring processes and assign to a team for monitoring them. It is crucial to understand the

LSA's architecture including technology stack, interfacing applications, and organize a demo of the application, and the use cases the prepare some sort of a document especially if documentation is unavailable or incomplete. This approach would be the Greenfield approach where besides the business logic, the new application is free of constraints of a pre-existing system. Some steps in this strategy include:

- Analyze the legacy application for the number of users, the number of interfaces with other applications, the criticality, and complexity of the application
- Document accurate and complete business and security rules of the application
- Determine and define the data structures in the legacy application
- Determine what needs to change and what needs to be maintained
- Perform gap analysis on the coverage of the legacy application and the replacement given the new technology of business logic changes.
- Prepare a project plan for the entire application, choose most used technology and third-party integration where possible, recruit qualified staff,
- Minimize disruption of critical applications relying on the legacy system by implementing the least critical components first,
- Use API driven development by creating incremental chunks as APIs,
- Identify and back up the LSA data for each component,
- Use test-driven Agile development and automate testing, where possible, ensure that the testing environment mirrors the production environment.
- Include weekly demo of working pieces as training and energy booster of the staff.

- Run the new application parallel to the old until the modernization is complete.

Relate the Outcome to the Theoretical Framework

In Chapter 2, the present researcher presented an extensive literature review on the need for finding a common LSA modernization strategy, while taking change management, disruptive innovation, and complexity theories into account.

Change management theory. As seen in Chapter 2, three steps of this theory are unfreezing, changing, and refreezing, derived from the process followed when preparing a meal from frozen supplies (Cummings et al., 2016; Hartzell, 2017; Petiprin, 2016). The emerged common strategy factors unfreezing by determining the need for change, documenting complete business processes and rules handled by the LSA, abstracting the LSA into independent components, as well as analyzing the LSA for its complexities, criticality, and the number of users and applications affected. The strategy factors the changing phase by following a project plan and using a project plan to develop the new application. Then, factors the refreezing by test-driven Agile development, implementing the least critical components first, and including training in every phase of the implementation.

Disruptive innovation theory. Disruptive innovation is innovation that not only improves a market but can overshoot the needs of consumers while responding to disruptive threats and cause a disruption in that market by displacing established competitors in the market with a less expensive and accessible version of a product ("Disruptive innovations theory," n.d.; King & Baatartogtokh, 2015). Such disruptions introduced by new competitors could happen as a result of new technology which is periodic and threatens to destroy organizations that rely heavily on LSAs not easily adaptable to new technology (Bakhit, 2016; Christensen, 1997). Disruptive innovation research has shown that IT leaders sometimes fail to modernize LSAs especially if customers have not demanded the new technology (Sandström et al., 2014). Also, Tantry et al. (2017) identified the challenges of software modernization, which include high costs, lengthy processes which could cause the modernization to be obsolete pre-delivery, as well as the need for the parallel running of the LSA and its replacement. Given the challenges involved in modernization or lack thereof, many SMSEOs could potentially easily become victims of disruptive innovation when newcomers or competitors push them out of the market due to a missed opportunity to modernize their LSA.

Replacing LSAs with new development has advantages like using new technology and the option to add new modules, as well as disadvantages such as long implementation time, high implementation cost, user training required, must include proper testing and running old and new applications parallel (Tantry et al., 2017). This researcher observed that the common strategy from the present research could become a disruptive innovation if many SMSEOs decide to use this strategy for LSA modernization. Even though Tantry et al. (2017) determined parallel running, training and proper testing as disadvantages of the replacement strategy, the present research showed that the emerged common strategy had those characteristics; however, 90% of the participants still preferred it.

Complexity theory. Restating this theory as when a system constitutes ongoing instability and entropy conditions and as a result, varying structures and patterns emerge as the system evolves into something new (Lowell, 2016). Reviewing literature in Chapter 2 exposed some existing LSA modernization strategies such as migration, maintenance, re-hosting as virtual machines, re-hosting on new hardware, replacement with new development, replacement with commercial-off-the-shelf (COTS), reengineering, wrapping, source code translation, retargeting, revamping, and evolution (Kuipers, 2002; Tantry et al., 2017; Zheng, 2013). In Round 1 of the present research six strategies emerged from the 13 participants some of which intersect with the existing strategies found in Chapter 2. Even though the data converged to three strategies by Round 2 and two strategies by Round 3 at a 90% convergence, the existence of the many strategies highlights the complex nature of the LSA modernization problem. By Round 3, although P11 has chosen Strategy two in both rounds two and three, P11 provided additional consideration stating that even though Strategy 2 was their most preferred they believe that Strategy 3 would have been equally good. Although this was not new information since Strategy 3 had been one of the strategies that P11 described in Round 1, this consideration exposed even more why finding a common strategy falls under complexity theory.

According to Nicas and Creswell (2019), the Boeing 737 Max plane is built upon a 1960s LSA of the original 737 and is, therefore, a legacy of its past, which contributed heavily to the recent plane crashes. Per Nicas and Creswell (2019), the pilots were comfortable flying the plane without the extra overhead of training, and redesigning proved to be cheaper and faster for Boeing than starting anew. The grounding of the Boeing 737 Max planes would amount to about a $1 to $5 billion loss (Isidore, 2019). This monetary loss, including the deadly crashes, may be worse than what it would have cost if modernization had been by replacing the LSA with new development as opposed to a redesign of the old software that was missing some vital crisis features (Isidore, 2019; Nicas & Creswell, 2019). While it is likely that organizations have used different processes for modernizing their LSAs and may have even abandoned some, the literature reviewed in Chapter 2 indicated the need for the present study as a step toward bridging the knowledge gap in the literature that warrants a common LSA modernization strategy which promotes innovation and covers change management. This researcher was able to identify an LSA modernization strategy that most SMSEOs can leverage to overcome their modernization challenges and salvage their indispensable business-critical LSAs, and therefore fulfilled the purpose of this qualitative Delphi study.

Contribution to Business Technical Problem

LSAs are large and complex applications that are critical for business but resist modification (Crotty & Horrocks, 2017; Srinivas et al., 2016). The LSAs are typically created using outdated technologies, although the applications remain indispensable to the organizations because of the daily business-critical use (Rai et al., 2015; Srinivas et al., 2016). The failure of an indispensable LSA can have a significant impact on business (Crotty & Horrocks, 2017; Srinivas et al., 2016). Typically with such indispensable LSAs, there is little or no documentation, and even when they exist, the documentation does not provide reliable information, and so it is hard to understand or update the systems see Table 8 in which a number of participants list this as a modernization challenge (Srinivas et al., 2016). LSAs pose some challenges to organizations such as slow speed, high maintenance costs and even costly fault detection due to obsolete technology; and for the organizations to modernize or refactor said applications, they need a Business Process Reengineering (BPR) (Srinivas et al., 2016). A BPR is a way to deal with change management where the related undertakings required to get a particular business result are fundamentally updated ("Business process reengineering (BPR)," n.d.). The general IT problem contemplated in the present study is the lack of common strategies for modernizing indispensable LSAs as technology changes (Crotty & Horrocks, 2017). The business IT problem was that many IT managers in SMSEO have no common strategy for modernizing their indispensable business-critical LSAs as technology changes at a fast pace, to match the technological changes and handle their business challenges, causing the modernization process to be inefficient and costly (Crotty & Horrocks, 2017; Morton et al., 2015; Rai et al., 2015). Adopting a common modernization strategy would help the indispensable LSAs to maintain their relevance and usability in the organization. (Jain & Chana, 2015). Having a common modernization strategy in place is beneficial to the decision-makers of SMSEOs because they would be able to continually provide services to both internal clients and paying customers while also leveraging modern technology as it changes (Bakhit, 2016; Vecchiato, 2017). With the study results indicating a convergence after only two rounds, it indicated that the challenges faced with LSA modernization are similar between the various IT leads. Therefore, the converged strategy Strategy 2: replacement with new development) along with its pros and cons would add value to the LSA, promote flexibility, and provide a fresh perspective for most IT leads in SMSEOs.

Recommendations for Further Research

As presented in Chapter 4, there were a few modifications to the data collection process that the present researcher planned in Chapter 3. Some of the modifications came as a result of lessons learned during the data collection process. From the data collection experience, some recommendations for future research include always providing a deadline and possibly an incentive for timely response, ask as many questions as possible in one sitting to avoid having to repeatedly contact the participants, and inform the participants about the purpose of collecting their contact information. In the present research, using the qualitative Delphi technique was convenient as the participants could take their time and present their experience in modernizing LSAs. However, it may be even more beneficial to organize a synchronous session to tally and present round results, while the participants are still on that topic. More recommendations include:

1. Use the list of strategies proposed in this qualitative Delphi study as the pre-selected list for a modified quantitative Delphi study with an even broader set of participants. Having a pre-selected list along with the option for the participants to add more or update the list as they see fit will give the participants an idea of how to or how not to structure their modernization strategies, even it is not in the pre-selected list. Also, as part of the quantitative survey test the factors that attribute a successful modernization strategy.
2. Study the relationship between the challenges that IT leads have faced in modernizing LSA with their choice of a modernization strategy.
3. Experimental research to compare the outcome of using Strategy 2 to that of Strategy 6.
4. What criteria can IT leads use to measure and evaluate the performance outcome of an LSA modernization strategy, what security measures to consider and how do the criteria relate to the role/ job title of the employee who suggested the strategy?

Conclusions

Using a three-round Delphi technique in this qualitative research, a panel of IT leads outlined and described the LSA modernization strategies they have used in the past or are currently using. The three most prevalent strategies were replacement with new development, SABSA, and the problem-solving oriented strategy, though the replacement with new development outshined the others at an 80% (replacement with new development) to 10% (SABSA), to 10% (problem-solving strategy). This conclusion was agreed upon by 90% of the participant, while 10% stuck to the fact that Strategy 6 was their preferred strategy all things considering. Given the vast majority, the present researcher concluded that Strategy 2 should be the common LSA modernization strategy. Strategy 2 is the replacement of LSA with new development by abstracting functions into independent components and creating microservices starting with the least critical component and assigning the components to Agile teams complete with the full development life cycle. The conclusion of this qualitative Delphi study aligns with the importance of LSA modernization in SMSEOs in helping them use new technology, innovate, and grow their business (Tantry et al., 2017). This conclusion also aligns with one of the opinions regarding the recent Boeing 737 Max crashes, which indicated that LSA modernizing with new development is best (Nicas & Creswell, 2019). This researcher was able to observe from IT leads of SMSEOs that the most preferred LSA modernization strategy is to replace with new development, and this strategy was initially suggested by IT leads with higher ranking such as CTO and later accepted by 90% of the participants.

REFERENCES

Adler, M., & Ziglio, E. (1996). *Gazing into the oracle: The Delphi method and its application to social policy and public health*. London: Jessica Kingsley Publishers.

Aengenheystera, S., Kerstin Cuhlsb, L., Heiskanen-Schüttlera, M., Huckd, J., & Muszynska, M. (2017). Real-Time Delphi in practice - A comparative analysis of existing software-based tools. *Technological Forecasting & Social Change, 118*(1), 15-27. doi:10.1016/j.techfore.2017.01.023

Aguirre, D., & Alpern, M. (2014). 10 Principles of leading change management. *Strategy and Business, 2014*(75), 1-8.

Ali, N. R., & Lai, R. (2015). A method of requirements change management for global software development. *Information and Software Technology, 70*, 49-67. doi:10.1016/j.infsof.2015.09.005

Aser, M. B. (2015). 6 Smoothed complexity theory. *ACM Trans. Comput. Theory, 7*(21). doi:10.1145/2656210

Bakhit, W. (2016). Impact of disruptive innovations on mobile telecom industry in lebanon. *International Journal of Research in Business and Social Science, 5*(3), 80-94. doi:10.20525/ijrbs.v5i3.437

Baruch, Y., & Holtom, B. C. (2008). Survey response rate levels and trends in organizational research. *human relations, 61*(8), 1139-1160. doi:10.1177/0018726708094863

Beijert, L. (2016). *Designating legacy status to IT systems: A framework in relation to a future-oriented perspective on legacy systems.* (Master's thesis), Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:hj:diva-30337

Bergner, N. M. v., & Lohmann, M. (2014). Future challenges for global tourism: A Delphi survey. *Journal of Travel Research, 53*(4), 420-432. doi:10.1177/0047287513506292

Bharathy, G. K., & McShane, M. K. (2014). Applying a systems model to enterprise risk management. *Engineering Management Journal, 26*(4).

Bowen, D. (2017). *A modified Delphi study of challenges archivists encounter adopting cloud storage for long-term digital preservation.* (Doctoral dissertation), Retrieved from ProQuest Dissertation and Theses database (10255839)

Business process reengineering (BPR). (n.d.). *TechTarget.* Retrieved from https://searchcio.techtarget.com/definition/business-process-reengineering

Christensen, C. (1997). *The innovator's dilemma: When new technologies cause great firms to fail*. Boston, MA: Harvard Business Press.

Cicmil, S., Cooke-Davies, T., Crawford, L., & Richardson, K. (2009). *Exploring the complexity of projects: Implications of complexity theory for project management practice*. Newtown Square, PA: Project Management Institute.

Complexity. (n.d.). *In Cambridge English Dictionary*. Retrieved from https://dictionary.cambridge.org/us/dictionary/english/complexity

Cooke-Davies, T. (2011). *Aspects of complexity: Managing projects in a complex world*. Project Management Institute.

Cooper, D., & Schindler, P. (2014). *Business research methods, 12th Edition*. New York, NY: McGraw-Hill/Irwin.

Creswell, J. W. (1998). *Qualitative inquiry and research design : Choosing among five traditions*. London: Thousand Oaks.

Crotty, J., & Horrocks, I. (2017). Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company. *Applied Computing and Informatics, 13*(2), 175-183. doi:10.1016/j.aci.2016.12.001

Cummings, S., Bridgman, T., & Brown, K. G. (2016). Unfreezing change as three steps: Rethinking Kurt Lewin's legacy for change management. *human relations, 69*(1), 33-60. doi:10.1177/0018726715577707

Dedeke, A. (2012). Improving legacy-system sustainability: A systematic approach. *IT Professional Magazine, 14*(1), 38-43. doi:10.1109/MITP.2012.10

Deschene, M. (2016). *Embracing security in all phases of the software development life cycle: A Delphi study*. (Doctoral dissertation), Retrieved from ProQuest Dissertation and Theses database (10156658)

Disruptive innovations theory. (n.d.). *Christensen Institute*. Retrieved from https://www.christenseninstitute.org/disruptive-innovations/

Dutcher, S. T. (2011). *Not on my watch: A qualitative study of the role of local law enforcement in terrorism prevention* (Doctoral dissertation), Retrieved from ProQuest Dissertation and Theses database (3481930)

EL Beggar, O., Bousetta, B., & Gadi, T. (2014). Getting objects methods and interactions by extracting business rules from legacy systems. *Journal of Systems Integration (1804-2724), 5*(3), 32-56.

El-Gazzar, R., Hustad, E., & Olsen, D. H. (2016). Understanding cloud computing adoption issues: A Delphi study approach. *The Journal of Systems and Software, 118*(C), 64-84. doi:10.1016/j.jss.2016.04.061

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

Fletcher, A. J., & Marchildon, G. P. (2014). Using the delphi method for qualitative, Participatory action research in health leadership. *International Journal of Qualitative Methods, 13*(1), 1-18.

Foster, J. L. (2011). *Leveraging convergent technologies: A review of the use and reuse of legacy technology.* (Doctoral dissertation), Retrieved from ProQuest Dissertation and Theses database (3443319)

Giles, R. A. (2015). *Complexity and information technology project failure: Application of an exploratory regression model.* (Doctoral dissertation), Retrieved from ProQuest Dissertations Publishing (3682574)

Guba, E. G., & Lincoln, Y. S. (1989). *Fourth Generation Evaluation*. Newbury Park, CA: Sage Publications.

Guest, G., Bunce, A., & Johnson, L. (1995). How many interviews are enough? An experiment with data saturation and variability and a good number of journals in the. *Morse Sandelowski Bluff Byrne*. doi:10.1177/1525822X05279903

Hakemi, A., Jeong, S. R., Ghani, I., & Sanaei, M. G. (2014). Enhancement of vector method by adapting octave for risk analysis in legacy systems. *KSII Transactions on Internet and Information Systems, 8*(6), 2118-2138. doi:10.3837/tiis.2014.06.018

Hartzell, S. (2017). Lewin's 3-stage model of change: Unfreezing, changing & refreezing - video & lesson transcript. *Study.com*. Retrieved from http://study.com/academy/lesson/lewins-3-stage-model-of-change-unfreezing-changing-refreezing.html

Her, P. P. (2017). *The lived experiences of Asian American women managers and microaggressions in the workplace.* (Doctoral dissertation), Retrieved from ProQuest Dissertation and Theses database (10262470)

Hsu, C.-C., & Sandford, B. A. (2007). The Delphi technique: Making sense of consensus. *Practical Assessment, Research & Evaluation, 12*(10), 1-8.

Hwang, B.-N., Huang, C.-Y., & Yang, C.-L. (2016). Determinants and their causal relationships affecting the adoption of cloud computing in science and technology institutions. *Innovation : Management, Policy & Practice, 18*(2), 164-190. doi:10.1080/14479338.2016.1203729

Isidore, C. (2019, March 13). Grounding all 737 Max planes could cost Boeing billions of dollars. *CNN Business*. Retrieved from https://www.cnn.com

Islam, M., Toma, T. R., Selim, M., Gias, A. U., & Khaled, S. M. (2016). Design migration from procedural to object oriented paradigm by clustering data call graph. *International Journal of Information Engineering and Electronic Business, 8*(2), 1-13. doi:10.5815/ijieeb.2016.02.01

Jain, S., & Chana, I. (2015, September). *Modernization of legacy systems: A generalised roadmap*. Paper presented at the Sixth International Conference on Computer and Communication Technology 2015, New York, NY.

Kaufman, S. A. (1993). *The origins of order: Self-organization and selection in evolution*. New York, NY: Oxford University Press.

Kaur, H., Ahamad, S., & Verma, G. N. (2016). Legacy program estimation. *International Journal of Computer Science and Information Security, 14*(2), 30-34.

Kehr, S., Quiñones, E., Böddeker, B., & Schäfer, G. (2015, June). *Parallel execution of AUTOSAR legacy applications on multicore ECUs with timed implicit communication.* Paper presented at the 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA.

King, A. A., & Baatartogtokh, B. (2015). How useful Is the theory of disruptive innovation? *MIT Sloan Management Review, 57*(1), 77-90.

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 EducationalTechnology, 44*(4), 536-543. doi:10.1111/bjet.12049

Kuipers, T. (2002). *Techniques for understanding legacy software systems.* (Doctoral dissertation), Universiteit van Amsterdam, Retrieved from Digital Production Center UBA (204294)

Labs, C. (Producer). (2014). Complexity science overview. *YouTube*. Retrieved from https://www.youtube.com/watch?v=r-JiHyejpn8

Labs, C. (Producer). (2017). Complexity theory short film. *YouTube*. Retrieved from https://www.youtube.com/watch?v=lfa752nng90

Lawrence, P. (2015). Leading Change – Insights Into How Leaders Actually Approach the Challenge of Complexity. *Journal of Change Management, 15*(3), 231-252. doi:10.1080/14697017.2015.1021271

Leech, N. L., & Onwuegbuzie, A. J. (2007). An array of qualitative data analysis tools: A call for data analysis triangulation. *School Psychology Quarterly*, *22*(4), 557-584. doi:10.1037/1045-3830.22.4.557

Leedy, P. D., & Ormrod, J. E. (2014). *Practical Research: Planning and Design* (K. Davis Ed. 11th Edition ed.).

Letier, E., Stefan, D., & Barr, E. T. (2014, May). *Uncertainty, risk, and information value in software requirements and architecture.* Paper presented at the 36th International Conference on Software Engineering, Hyderbad, India.

Linstone, H. A., & Turoff, M. (2011). Delphi: A brief look backward and forward. *Technological Forecasting and Social Change, 78*(8), 1712-1719. doi:10.1016/j.techfore.2010.09.011

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of Atmospheric Sciences, 20*, 130-141. doi:10.1175/1520-0469(1963)0202.0.CO;2

Lowell, K. R. (2016). An application of complexity theory for guiding organizational change. *The Psychologist-Manager Journal, 19*(3-4), 148-181. doi:10.1037/mgr0000044

Luftman, J., & Kempaiah, R. (2007). An update on business-it alignment: “A line" has been drawn. *MIS Quarterly Executive, 6*(3), 165-177.

Marcella, M., & Rowley, S. (2015). An exploration of the extent to which project management tools and techniques can be applied across creative industries through a study of their application in the fashion industry in the North East of Scotland. doi:10.1016/j.ijproman.2014.12.002

Marfatia, M. (2014). How legacy code is exposing business and government systems. *Security InfoWatch*.

Mason, M. (2010). Sample size and saturation in phd studies using qualitative interviews. *Forum: Qualitative Social Research. Sep2010, 11*(3), 1.

Maxey, D., & Kezar, A. (2016). Leveraging the delphi technique to enrich knowledge and engage educational policy problems. *Educational Policy, 30*(7), 1042-1070. doi:10.1177/0895904815586856

McMillan, S. S., Kelly, F., Sav, A., Kendall, E., King, M. A., Whitty, J. A., & Wheeler, A. J. (2014). Using the nominal group technique: How to analyse across multiple groups. *Health Services and Outcomes Research Methodology*, *14*(3), 92-108. doi:10.1007/s10742-014-0121-1

McMillan, S. S., King, M., & Tully, M. P. (2016). How to use the nominal group and Delphi techniques. *Int J Clin Pharm, 38*(1), 655-662. doi:10.1007/s11096-016-0257-x

Migration. (n.d.). *SearchCIO*. Retrieved from http://searchcio.techtarget.com/definition/migration

Morton, J., Beckford, J., & Cooke, L. (2015). Towards a Rosetta Stone for translating data between information systems. *Business Information Review, 32*(4), 220-230. doi:10.1177_0266382115616235

National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. (1979). *Belmont report: Ethical guidelines for the protection of human subjects of research*. Retrieved from https://www.hhs.gov/ohrp/regulations-and-policy/belmont-report

Nicas, J., & Creswell, J. (2019, April 08). Boeing's 737 Max: 1960s Design, 1990s Computing Power and Paper Manuals. *The New York Times*. Retrieved from https://www.nytimes.com

Norfolk, D. (2014). Legacy modernisation. *Bloor Research*. Retrieved from https://www.bloorresearch.com/issue/legacy-modernisation/

Oliveira, T., Thomas, M., & Espadanal, M. (2014). Assessing the determinants of cloud computing adoption: An analysis of the manufacturing and services sectors. *Information & Management*, *51*(5), 497-510. doi:10.1016/j.im.2014.03.006

OnBase. (n.d.). The trouble with legacy systems an insurance executive's challenges and options. Retrieved from https://www.onbase.com/~/media/Files/hyland/whitepaper/wp_trouble-with-legacy-systems.pdf

Petiprin, A. (2016). Lewin's change theory. *Nursing Theory*. Retrieved from http://www.nursing-theory.org/theories-and-models/Lewin-Change-Theory.php

Populations and sampling. (n.d.). Retrieved from http://www.umsl.edu/~lindquists/sample.html

Quezada, N. (2017). The 4 types of software maintenance. *Endertech*. Retrieved from https://endertech.com/blog/maintenance-bug-fixing-4-types-maintenance

Radhakrishnan, H., Rouson, D. W. I., Morris, K., Shende, S., & Kassinos, S. C. (2015). Using coarrays to parallelize legacy fortran applications: Strategy and case study. *Scientific Programming, 2015*, 1-12. doi:10.1155/2015/904983

Rai, R., Sahoo, G., & Mehfuz, S. (2015). Exploring the factors influencing the cloud computing adoption: a systematic study on cloud migration. *SpringerPlus, 4*(1), 197. doi:10.1186/s40064-015-0962-2

Razavian, M., & Lago, P. (2015). A systematic literature review on SOA migration. *Journal of Software: Evolution & Process, 27*(5), 337-372. doi:10.1002/smr.1712

Roberts, C. M. (2010). *The dissertation journey: A practical and comprehensive guide to planning, writing, and defending your dissertation*: Corwin Press.

Rowe, G., & Wright, G. (1999). The Delphi technique as a forecasting tool: Issues and analysis. *International Journal of Forecasting, 15*(4), 353-375. doi:10.1016/S0169-2070(99)00018-7

Ruan, W., Vyas, T., Liu, Y., & Spear, M. (2014, February). *Transactionalizing legacy code: An experience report using GCC and memcached.* Paper presented at the ASPLOS ‘14, New York, NY.

Sadeghi, A.-R., Davi, L., & Larsen, P. (2015, April). *Securing legacy software against real-world code-reuse exploits: Utopia, alchemy, or possible future?* Paper presented at the 10th ACM Symposium on Information, Computer and Communications Security, New York, NY.

Saini, M., Mehmi, S., & Chahal, K. K. (2016). Understanding open source software evolution using fuzzy data mining algorithm for time series data. *Hindawi Publishing Corporation Advances in Fuzzy Systems, 2016*. doi:10.1155/2016/1479692

Sandström, C., Berglund, H., & Magnusson, M. (2014). Symmetric assumptions in the theory of disruptive innovation: Theoretical and managerial implications. *Creativity and Innovation Management, 23*(4), 472-483. doi:10.1111/caim.12092

Sargut, G. (2011). Learning to live with complexity. *Harvard business review.*

Saynisch, M. (2010a). Beyond frontiers of traditional project management: An approach to evolutionary, self-organizational principles and the complexity theory-results of the research program. *Project Management Journal, 41*(2), 21-37. doi:10.1002/pmj.20159

Saynisch, M. (2010b). Mastering complexity and changes in projects, economy, and society via Project Management Second Order (PM-2). *Project Management Journal. Dec2010, 41*(5), 4. doi:10.1002/pmj.20167

Serrano, N., Hernantes, J., & Gallardo, G. (2014). Service-Oriented architecture and legacy systems. *IEEE Software, 31*(5), 15-19. doi:10.1109/MS.2014.125

Shah, A. (2017). *Determination of isra framework using delphi methodology for small and midsized enterprises.* (Doctoral dissertation), Retrieved from ProQuest Dissertation and Theses database (10258389)

She, S., Ryssel, U., Andersen, N., Wąsowski, A., & Czarnecki, K. (2014). Efficient synthesis of feature models. *Information and Software Technology, 56*(9), 1122-1143. doi:10.1016/j.infsof.2014.01.012

Skulmoski, G. J., Hartman, F. T., & Krahn, J. (2007). The Delphi method for graduate research. *Journal of Information Technology Education, 6*.

Software maintenance. (n.d.). *The Economic Times.* Retrieved from https://economictimes.indiatimes.com/definition/software-maintenance

Software maintenance overview. (n.d.). *Tutorialspoint.com.* Retrieved from https://www.tutorialspoint.com/software_engineering/software_maintenance_overview.htm

Srinivas, M., Ramakrishna, G., Rao, K. R., & Suresh Babu, E. (2016). Analysis of legacy system in software application development: A comparative survey. *International Journal of Electrical and Computer Engineering (IJECE), 6*(1), 292-297. doi:10.11591/ijece.v6i1.8367

Stamford, C. (2014). Gartner says by 2016, the Impact of cloud and emergence of postmodern ERP will relegate highly customized ERP systems to [Press release]. Retrieved from http://www.gartner.com/newsroom/id/2658415

Strategy innovation. (n.d.). *Financial Times Lexicon.* Retrieved from http://lexicon.ft.com/Term?term=strategy-innovation

Tantry, H. S., Murulidhar, N. N., & Chandrasekaran, K. (2017). Implications of legacy software system modernization - A survey in a changed scenario. *International Journal of Advanced Research in Computer Science, 8*(7), 1002-1008. doi:10.26483/ijarcs.v8i7.4556

Transactionalize. (n.d.). *Urban Dictionary.* Retrieved from https://www.urbandictionary.com/define.php?term=Transactionalized

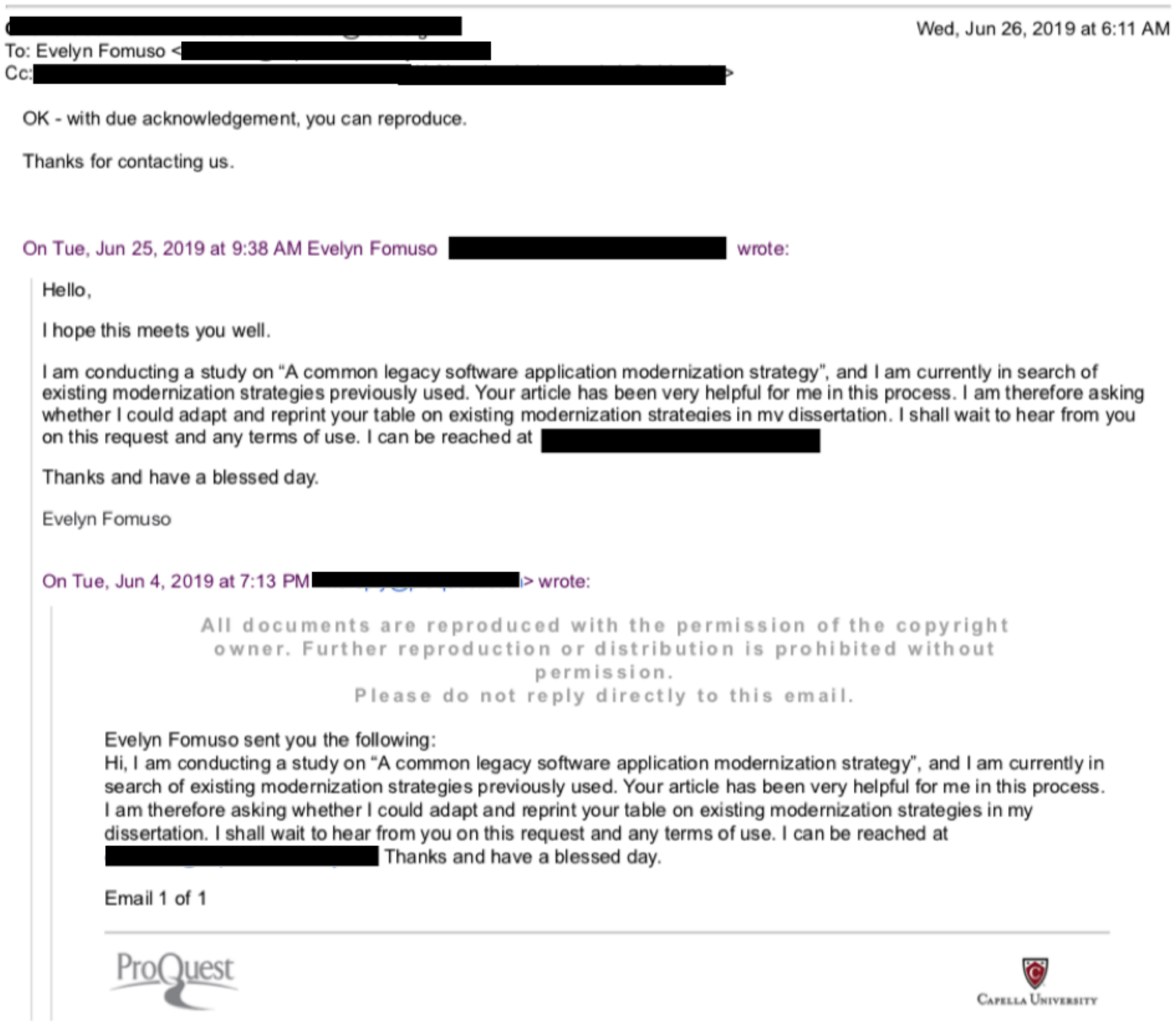
Vecchiato, R. (2017). Disruptive innovation, managerial cognition, and technology competition outcomes. *Technological Forecasting & Social Change, 116*, 116-128. doi:10.1016/j.techfore.2016.10.068

Vogel-Heuser, B., Fay, A., Schaefer, I., & Tichy, M. (2015). Evolution of software in automated production systems: Challenges and research directions. *Journal of Systems and Software, 110*, 54-84. doi:10.1016/j.jss.2015.08.026

Zanotta, D. (2013). *Managing complexity in virtual project teams: Understanding the lived experiences of the traditional project manager through phenomenological research.* (Doctoral dissertation), Retrieved from ProQuest Dissertation and Theses database (3590343)

Zheng, S. (2013). *An approach to implementing cloud service oriented legacy application evolution.* (Doctoral dissertation), Retrieved from https://www.dora.dmu.ac.uk/handle/2086/9657

APPENDIX A. PERMIT TO ADAPT AND REPRINT TABLE



[redacted] Wed, Jun 26, 2019 at 6:11 AM
To: Evelyn Fomuso <[redacted]
Cc: [redacted]

OK - with due acknowledgement, you can reproduce.

Thanks for contacting us.

On Tue, Jun 25, 2019 at 9:38 AM Evelyn Fomuso [redacted] wrote:

Hello,

I hope this meets you well.

I am conducting a study on "A common legacy software application modernization strategy", and I am currently in search of existing modernization strategies previously used. Your article has been very helpful for me in this process. I am therefore asking whether I could adapt and reprint your table on existing modernization strategies in my dissertation. I shall wait to hear from you on this request and any terms of use. I can be reached at [redacted]

Thanks and have a blessed day.

Evelyn Fomuso

On Tue, Jun 4, 2019 at 7:13 PM [redacted]> wrote:

All documents are reproduced with the permission of the copyright owner. Further reproduction or distribution is prohibited without permission.
Please do not reply directly to this email.

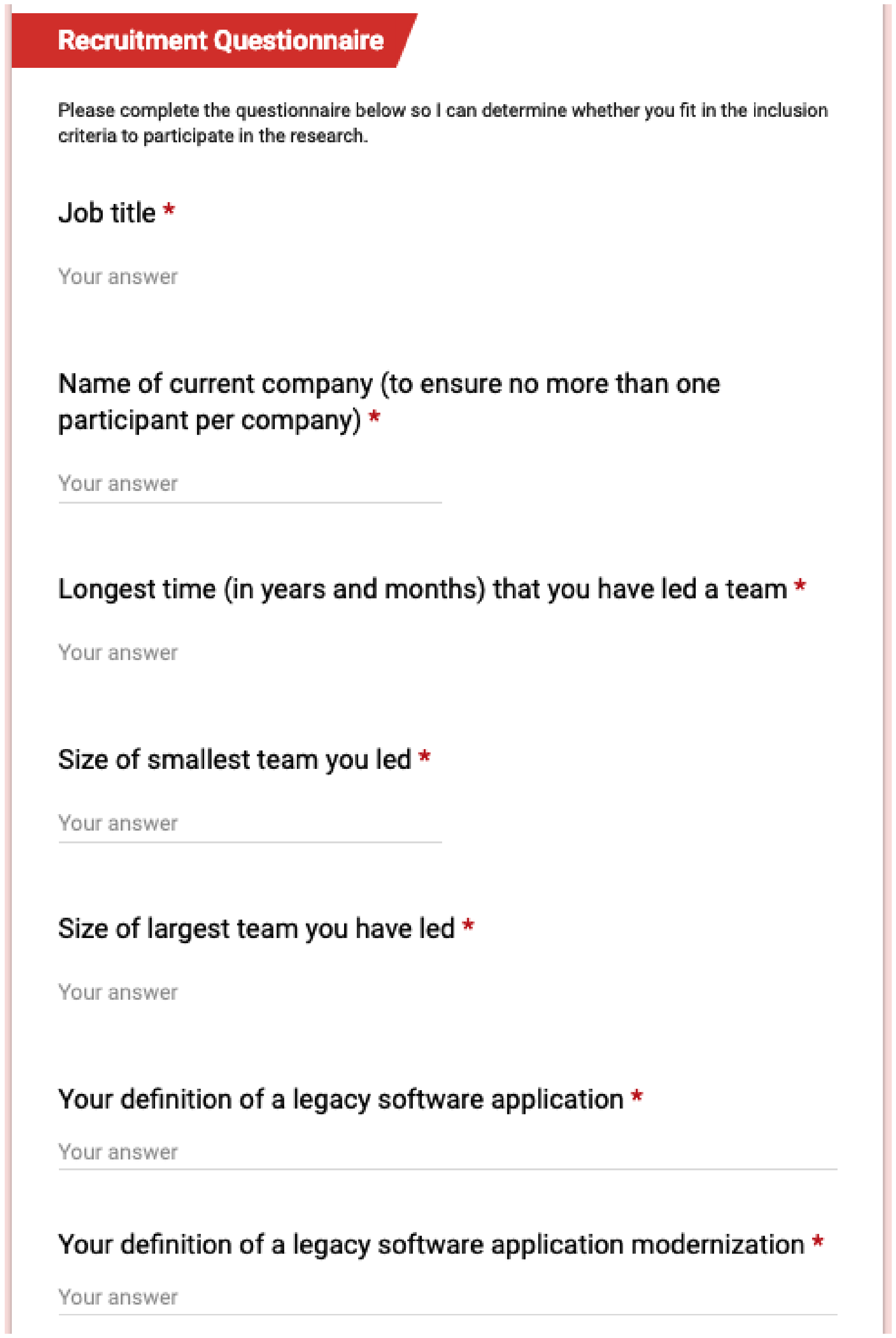
Evelyn Fomuso sent you the following:
Hi, I am conducting a study on "A common legacy software application modernization strategy", and I am currently in search of existing modernization strategies previously used. Your article has been very helpful for me in this process. I am therefore asking whether I could adapt and reprint your table on existing modernization strategies in my dissertation. I shall wait to hear from you on this request and any terms of use. I can be reached at [redacted] Thanks and have a blessed day.

Email 1 of 1

ProQuest

Capella University

APPENDIX B. RESEARCHER-DESIGNED RECRUITMENT QUESTIONNAIRE AND ROUND 1



Recruitment Questionnaire

Please complete the questionnaire below so I can determine whether you fit in the inclusion criteria to participate in the research.

Job title *

Your answer

Name of current company (to ensure no more than one participant per company) *

Your answer

Longest time (in years and months) that you have led a team *

Your answer

Size of smallest team you led *

Your answer

Size of largest team you have led *

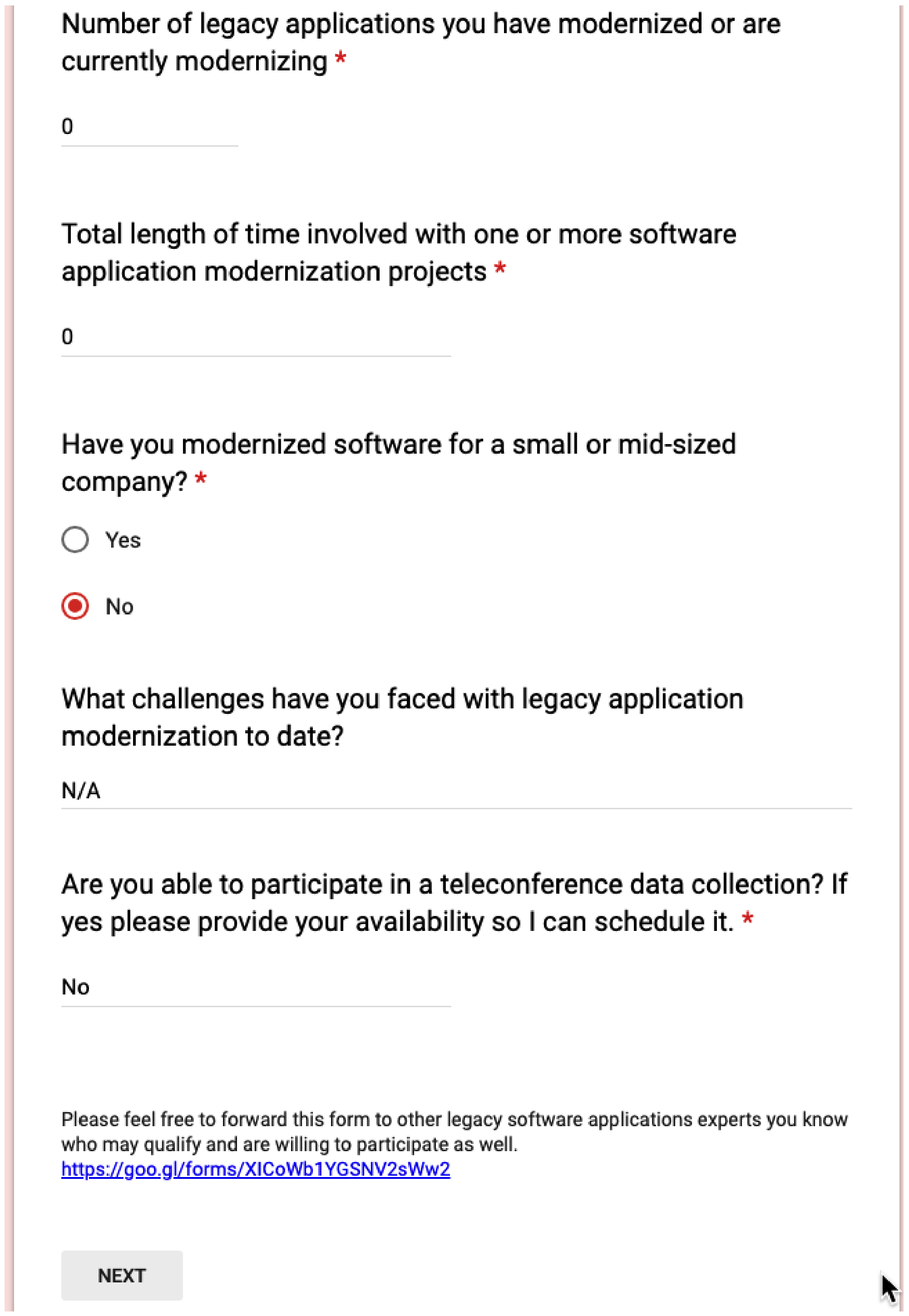
Your answer

Your definition of a legacy software application *

Your answer

Your definition of a legacy software application modernization *

Your answer



Number of legacy applications you have modernized or are currently modernizing *

0

Total length of time involved with one or more software application modernization projects *

0

Have you modernized software for a small or mid-sized company? *

Yes [✓ No]

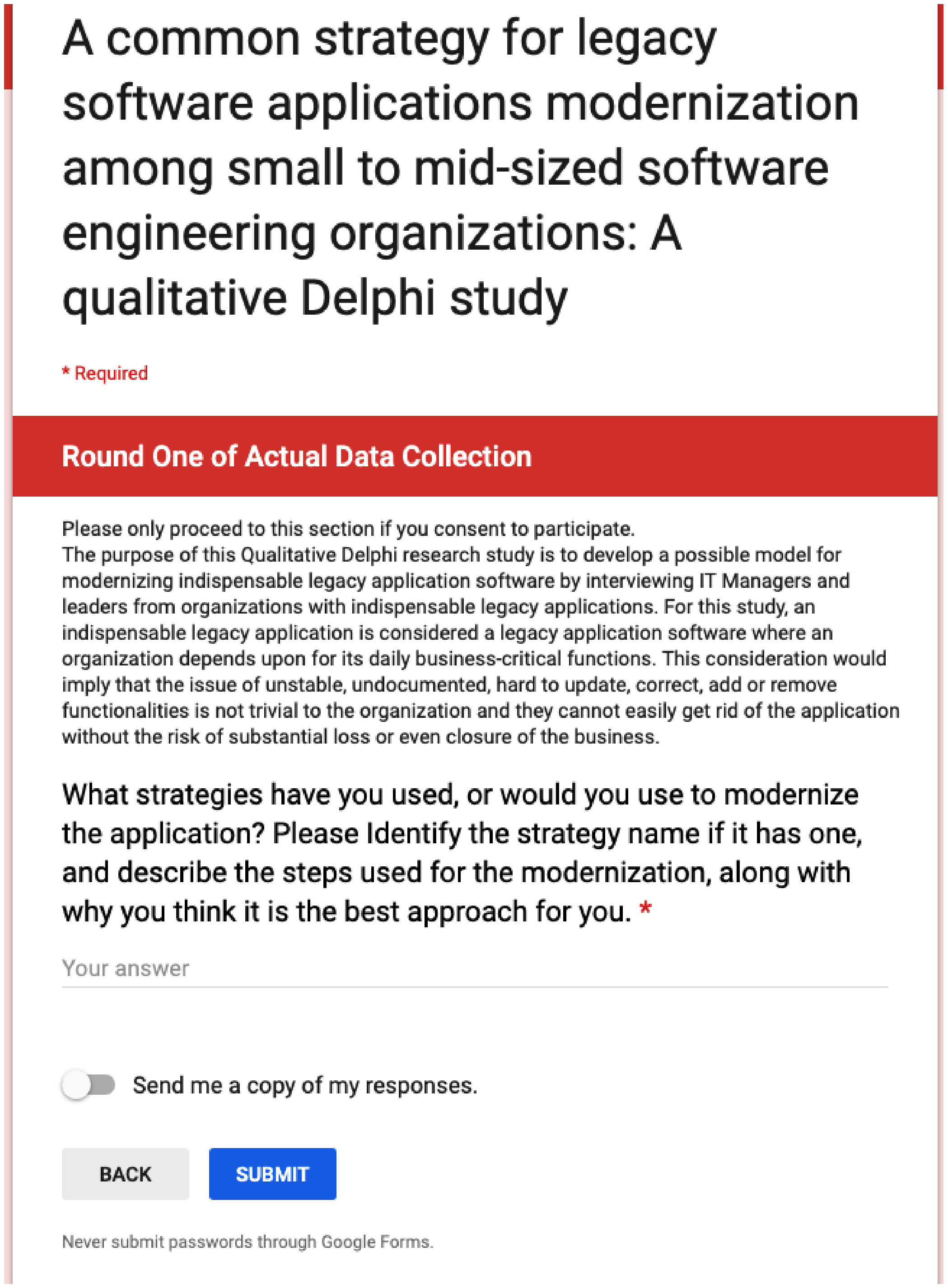
What challenges have you faced with legacy application modernization to date?

N/A

Are you able to participate in a teleconference data collection? If yes please provide your availability so I can schedule it. *

No

Please feel free to forward this form to other legacy software applications experts you know who may qualify and are willing to participate as well.
https://goo.gl/forms/XICoWb1YGSNV2sWw2


A common strategy for legacy software applications modernization among small to mid-sized software engineering organizations: A qualitative Delphi study

* Required

Round One of Actual Data Collection

Please only proceed to this section if you consent to participate.
The purpose of this Qualitative Delphi research study is to develop a possible model for modernizing indispensable legacy application software by interviewing IT Managers and leaders from organizations with indispensable legacy applications. For this study, an indispensable legacy application is considered a legacy application software where an organization depends upon for its daily business-critical functions. This consideration would imply that the issue of unstable, undocumented, hard to update, correct, add or remove functionalities is not trivial to the organization and they cannot easily get rid of the application without the risk of substantial loss or even closure of the business.

What strategies have you used, or would you use to modernize the application? Please Identify the strategy name if it has one, and describe the steps used for the modernization, along with why you think it is the best approach for you. *

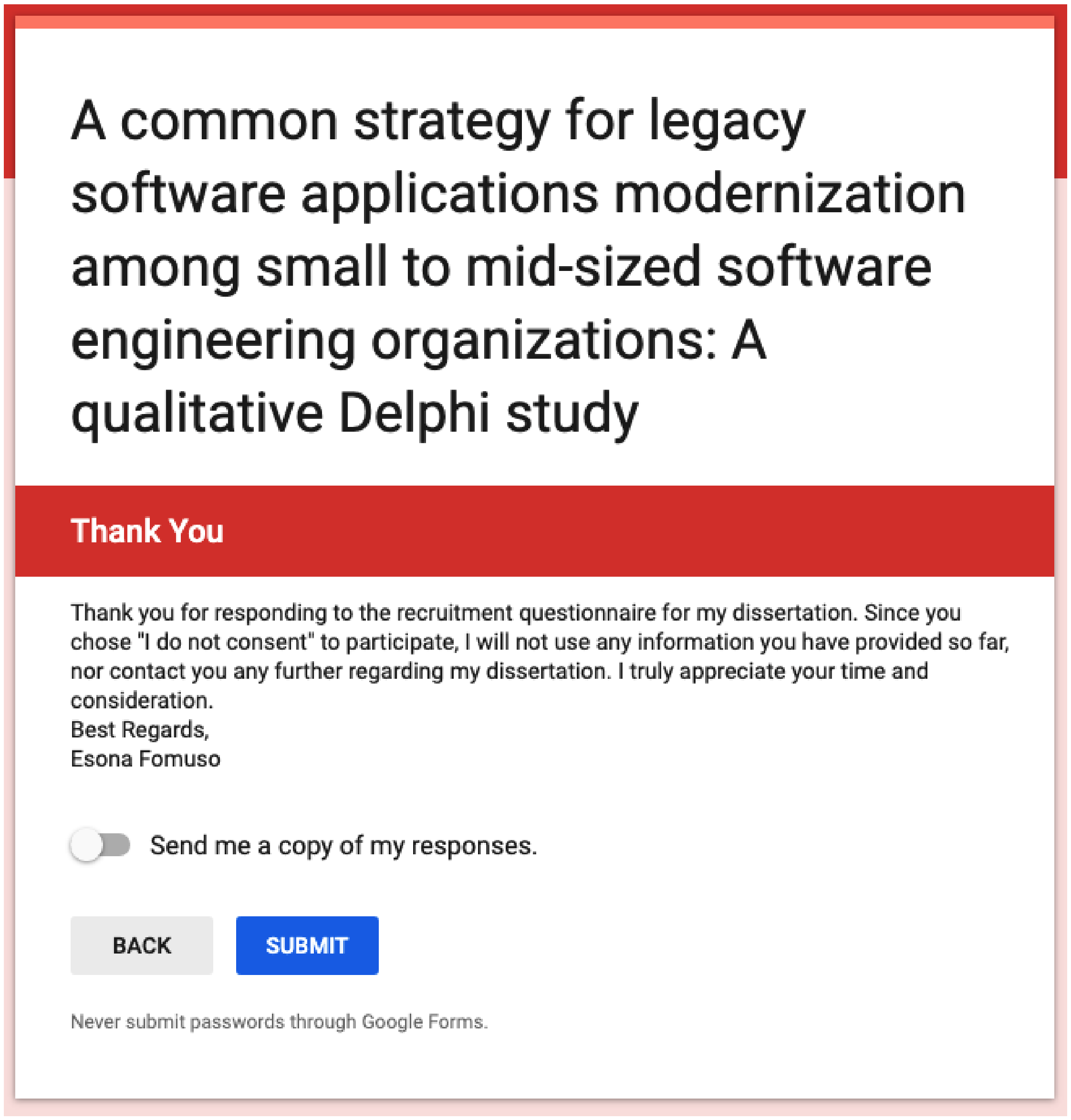
Your answer

Send me a copy of my responses.

BACK SUBMIT

Never submit passwords through Google Forms.

APPENDIX C. TRANSITION TO EXIT ON DO NOT CONSENT



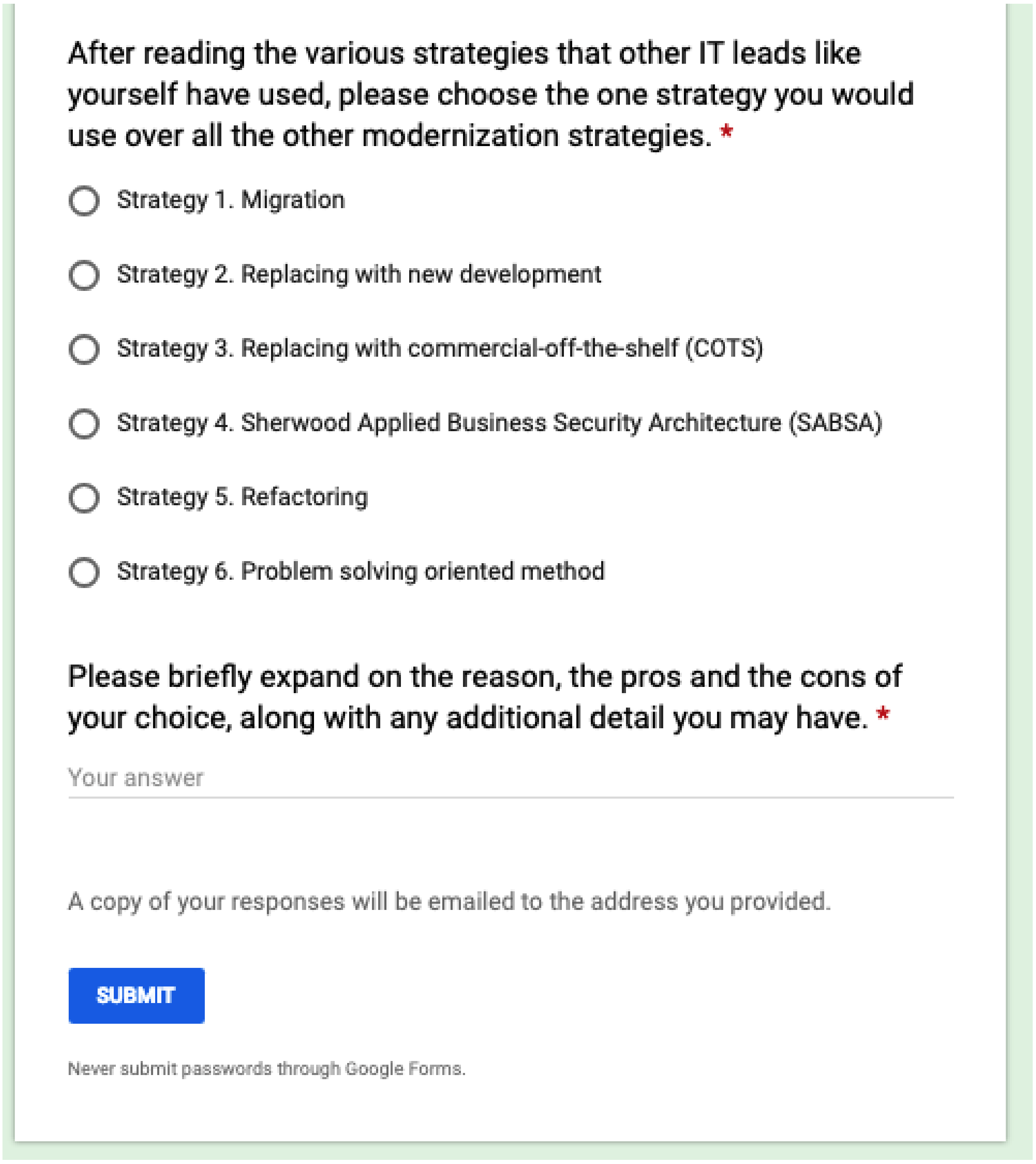
A common strategy for legacy software applications modernization among small to mid-sized software engineering organizations: A qualitative Delphi study

Thank You

Thank you for responding to the recruitment questionnaire for my dissertation. Since you chose "I do not consent" to participate, I will not use any information you have provided so far, nor contact you any further regarding my dissertation. I truly appreciate your time and consideration.
Best Regards,
Esona Fomuso

Send me a copy of my responses.
BACK
SUBMIT
Never submit passwords through Google Forms.

APPENDIX D. RESEACHER-DESIGNED ROUND 2 QUESTIONNAIRE



After reading the various strategies that other IT leads like yourself have used, please choose the one strategy you would use over all the other modernization strategies. *

- ○ Strategy 1. Migration
- ○ Strategy 2. Replacing with new development
- ○ Strategy 3. Replacing with commercial-off-the-shelf (COTS)
- ○ Strategy 4. Sherwood Applied Business Security Architecture (SABSA)
- ○ Strategy 5. Refactoring
- ○ Strategy 6. Problem solving oriented method

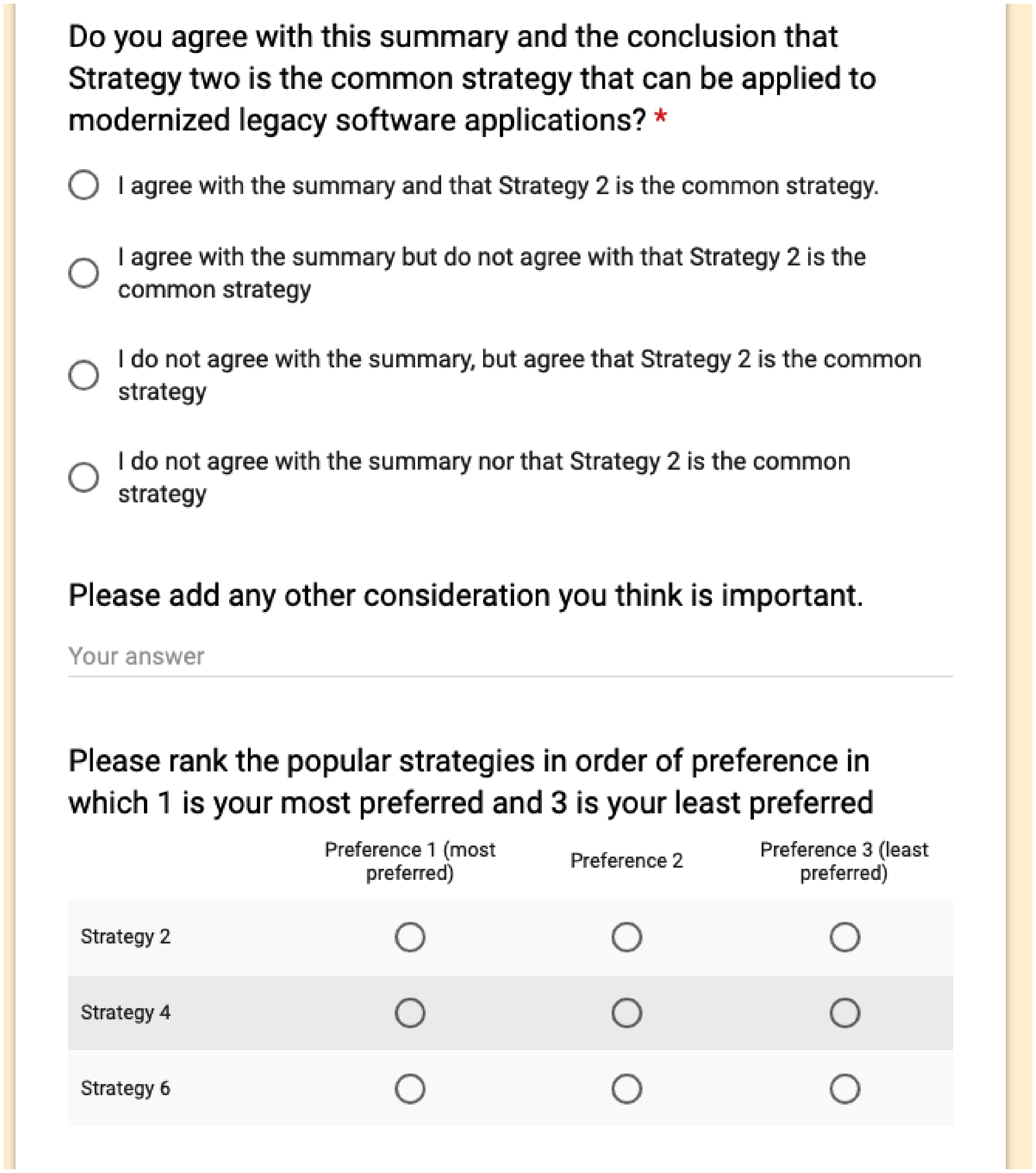
Please briefly expand on the reason, the pros and the cons of your choice, along with any additional detail you may have. *

Your answer

A copy of your responses will be emailed to the address you provided.

Never submit passwords through Google Forms.

APPENDIX E. RESEACHER-DESIGNED ROUND 3 QUESTIONNAIRE



Do you agree with this summary and the conclusion that Strategy two is the common strategy that can be applied to modernized legacy software applications? *

○ I agree with the summary and that Strategy 2 is the common strategy.

○ I agree with the summary but do not agree with that Strategy 2 is the common strategy

○ I do not agree with the summary, but agree that Strategy 2 is the common strategy

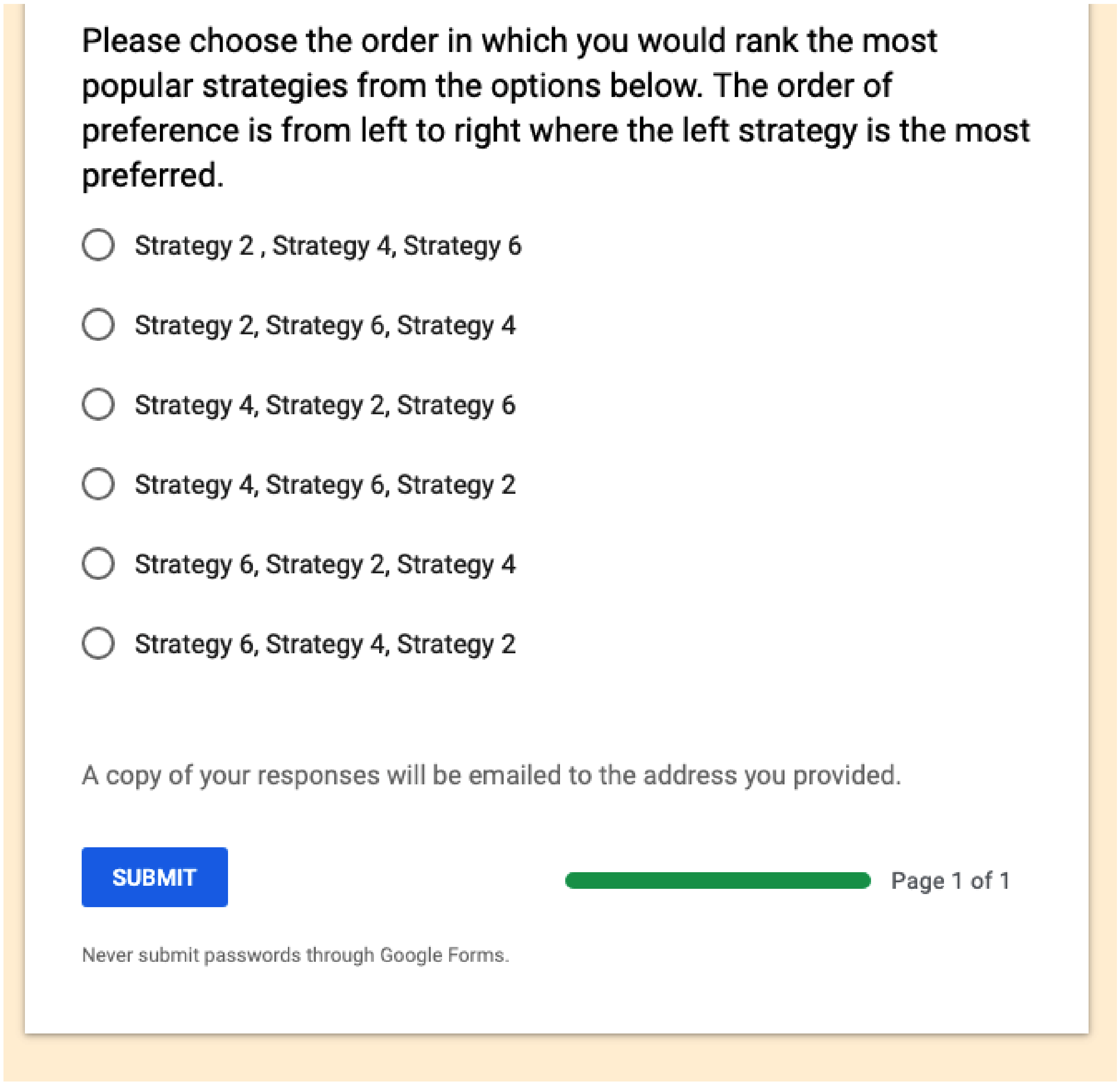
○ I do not agree with the summary nor that Strategy 2 is the common strategy

Please add any other consideration you think is important.

Your answer

Please rank the popular strategies in order of preference in which 1 is your most preferred and 3 is your least preferred

	Preference 1 (most preferred)	Preference 2	Preference 3 (least preferred)
Strategy 2	○	○	○
Strategy 4	○	○	○
Strategy 6	○	○	○



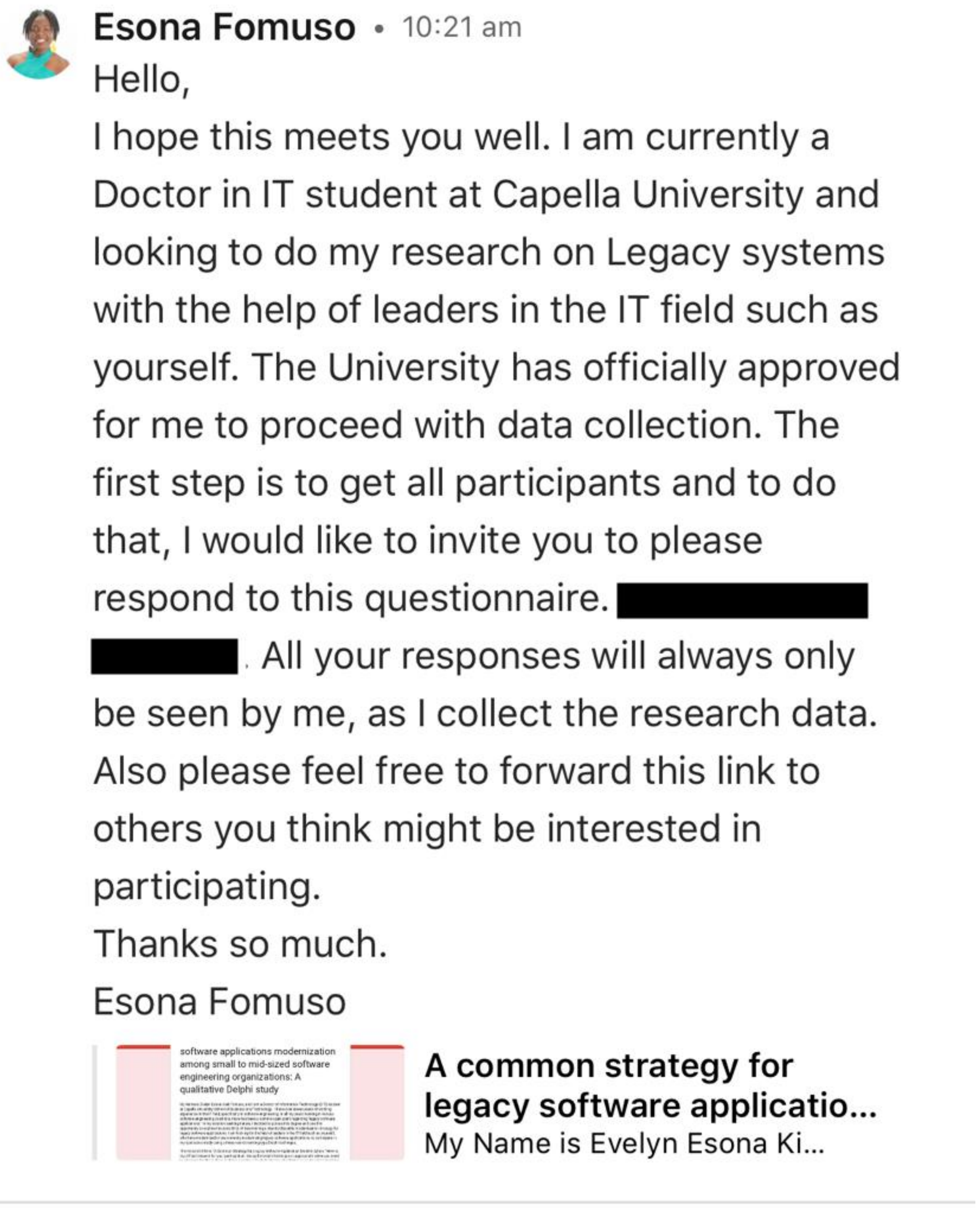
Please choose the order in which you would rank the most popular strategies from the options below. The order of preference is from left to right where the left strategy is the most preferred.

- ○ Strategy 2 , Strategy 4, Strategy 6
- ○ Strategy 2, Strategy 6, Strategy 4
- ○ Strategy 4, Strategy 2, Strategy 6
- ○ Strategy 4, Strategy 6, Strategy 2
- ○ Strategy 6, Strategy 2, Strategy 4
- ○ Strategy 6, Strategy 4, Strategy 2

A copy of your responses will be emailed to the address you provided.

Never submit passwords through Google Forms.

APPENDIX F. PARTICIPATION SOLICITATION EMAIL



Esona Fomuso • 10:21 am

Hello,

I hope this meets you well. I am currently a Doctor in IT student at Capella University and looking to do my research on Legacy systems with the help of leaders in the IT field such as yourself. The University has officially approved for me to proceed with data collection. The first step is to get all participants and to do that, I would like to invite you to please respond to this questionnaire. [redacted] [redacted]. All your responses will always only be seen by me, as I collect the research data. Also please feel free to forward this link to others you think might be interested in participating.

Thanks so much.

Esona Fomuso

software applications modernization among small to mid-sized software engineering organizations: A qualitative Delphi study

A common strategy for legacy software applicatio...

My Name is Evelyn Esona Ki...