A Value Model-Based Decision Framework for an Enterprise Data Architecture

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Kelly Fitzpatrick
Bachelor of Science
University of Arizona, 2015

Director: Dr. Paulo C. G. Costa, Associate Professor
Department of Systems Engineering and Operations Research

Fall Semester 2021
George Mason University
Fairfax, VA

Dedication

I dedicate this thesis to my family and friends who helped me, comforted me, and encouraged me to keep pushing myself to advance my education and accomplish this goal.

I want to thank my family for being understanding of the missed holidays and time I couldn't spend with them while I was working on this thesis. Thank you Dad, Mom, Michael, Max, and Wolf for cheering me on. Thank you for never doubting I could do this and all the heartening words you provided me. I know I made you proud.

I want to thank my boyfriend Jericho Quijada who has supported and encouraged me everyday. Thank you Jericho for being my daily reassurance and getting me through all the stressful days. I could not have done this with out your unwavering belief in me and the strength you provided me.

Thank you to my cat Apollo Fitzpatrick. You were my daily sunshine for the last 11 years. I appreciated the jolly happiness you brought into my life. I dedicate this to you.

Acknowledgments

I would like to thank the following people who made this possible.

Thank you to Paulo Costa, PhD my advisor for challenging me to pursue a masters thesis and knowing I could do this. Thank you for your endorsement and motivating me to proceed along with my education.

Thank you to my committee members Peggy S. Brouse, PhD and Syed Abbas K Zaidi, PhD for providing feedback and encouraging me to continue to working on this topic.

I want to thank my friends Elise Beisecker, PhD and Shanda Wilson, JD for providing career guidance and inspiring me to advance my education. I am grateful to have you as examples to aspire to and rely on in my life.

I want to thank Michael McCabe and Chris Sutton for helping me to get the scholarship to go back to school and championing for me. Thank you for making this possible.

Thank you to all who responded to my survey. The information you provided helped me to complete this paper. I appreciate your time and thought provoking answers.

Table of Contents

List of Tables

List of Figures

Abstract

A VALUE MODEL-BASED DECISION FRAMEWORK FOR AN ENTERPRISE DATA ARCHITECTURE

Kelly Fitzpatrick

George Mason University, 2021

Thesis Director: Dr. Paulo C. G. Costa

The objective of this paper is to improve an organization's decision-making process for developing an enterprise data architecture.

Enterprise data architectures are complex systems with multiple users who have competing objectives for the system. When developing an enterprise data architecture, it can be hard to satisfy all the organizations goals and maintain a consistent development plan.

The development team building the enterprise data architecture needs a robust decision-making process to help them determine the architecture design for the organization. The process needs to be able to quickly support questions that arise throughout the development of the system. The goal is to reduce the amount of time answering questions after a decision has been made on the architecture design and development plan.

The solution presented is a framework that uses value focused decision analysis and model-based systems engineering to develop a parametric model which can be used to identify the best architecture design for the organization and support answering questions the organization's leadership might ask [1].

Chapter 1: Introduction

Thesis Overview This paper is a demonstration of mastery for a Systems Engineering Masters of Science Degree. This thesis is an application of system engineering theories and methodologies applied to decision making when designing an enterprise data architecture.

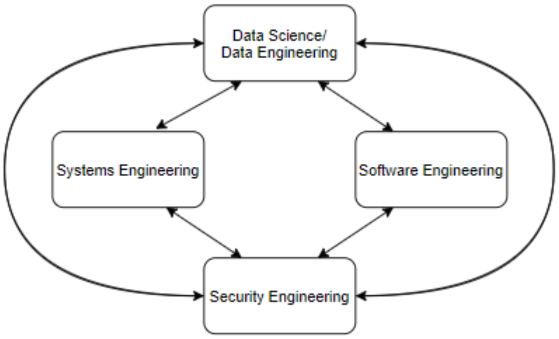


Figure 1.1: Four Engineering Fields

Systems Engineering, Security Engineering, Software Engineering, and Data Science are the four broad fields being applied to the topic of this paper. One of the main purposes for building an enterprise data architecture is to support the field of data science and data engineering [2]. Data Science and software engineering have been researching big data environments and their applications for over 10 years [3]. Security Engineering research has been focused on data security and mitigating security threats for targets like big data architectures [4]. The systems engineering field however has not published much research on enterprise data architectures [5]. This paper will be an example of how systems engineering can be applied to solving problems in this topic area.

Complex applications in the information technology field such as an enterprise data architectures can benefit from using more operations research and model-based systems engineering techniques to solve problems [6]. Before the development phase of an enterprise level application begins a systems engineer should derive essential functions of the system, validate system designs against the organization's requirements, develop end-to-end processes, and generate test plans [7]. These artifacts are important for ensuring a complex system is built properly. As security threats become more sophisticated and information technology systems become more complex, it is necessary to use Systems Engineering to help define and document risks within the system that need to be mitigated. Applying a more rigorous approach to the development and design of complex systems using systems engineering methods will help identify security vulnerability's early on which can be mitigated during the development process [8]. Using a systems engineering techniques will help organizations successfully develop their enterprise data architectures [9].

Reason for Topic Choice The topic of enterprise data architecture was chosen because of previous background knowledge in both the software engineering and data science fields and an awareness of current challenges in building data architectures. Data security is of particular interest because of it's importance and the design challenges it brings to data architectures.

Enterprise data architecture is a difficult topic because it spans multiple fields of engineering [10]. That challenge is what makes enterprise data architectures a good subject for systems engineers to study because systems engineers work to bring together multiple disciplines to solve problems [7]. The original intent was to study design patterns for data security and their effects on performance of the architecture, but as research went forward the problem focus evolved into supporting the challenges of decision making in general on an enterprise data architecture. Data security is a specific trade-off decision that this paper's framework could support in the future. Because there are so many different fields involved with developing an enterprise data architecture and so many sub-problems to be discussed, having a decision model to support data driven decision making was the first

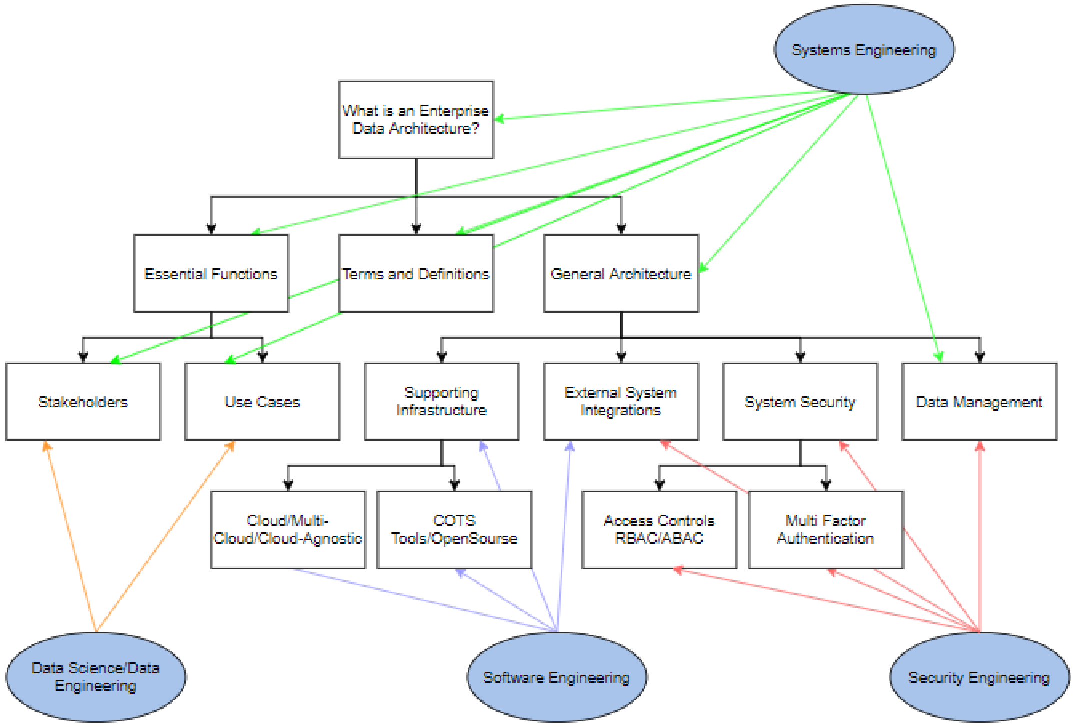

Systems Engineering
What is an Enterprise Data Architecture?
Essential Functions
Terms and Definitions
General Architecture
Stakeholders
Use Cases
Supporting Infrastructure
External System Integrations
System Security
Data Management
Cloud/Multi-Cloud/Cloud-Agnostic
COTS Tools/OpenSourse
Access Controls RBAC/ABAC
Multi Factor Authentication
Data Science/Data Engineering
Software Engineering
Security Engineering

Figure 1.2: Mapping of the Engineering Fields

step in supporting future trade-off analysis discussions.

1.1 Background

Overview Commercial Companies, Government, Academia, and other industries are all looking to maximize use of the data they have collected [3]. Data has become a valuable commodity which organizations can find multiple uses to benefit from. Many organizations are looking to acquire more data and to take advantage of the data they already have. This is why organizations are looking to create an enterprise data architecture that can help them organize data and free it from stove-piped applications where it can be analyzed with other sources of data [11].

Having organized data is the foundation for supporting other advanced efforts like artificial intelligence, streaming analytics, simulations, dashboarding etc [12]. Many of these efforts require multiple data sources to be cleaned, integrated, and stored in a place where users can access them [13]. An enterprise data architecture helps by centralizing storage of data, automating access, and making data sources discoverable to users within the organization [14].

There are several security risks to consider when developing an enterprise data architecture [15]. Organizations contain sensitive information they want to protect. Sensitive data could be financial data, personally identifiable information, proprietary information, customer data, contracts, security logs, etc. The enterprise data architecture should be designed in such a way that manages permissions and access to sensitive data [16]. There are several methods in use today for implementing access controls on applications and even more complex data architectures, but each of them have limitations that should be considered [8]. Choosing an implementation method depends on the risk posture of the organization and the amount of control they need on their data.

The goal of an enterprise architecture is to provide access to as much data as possible while limiting access to the sensitive parts of the data. To build an enterprise data architecture that meets the needs of an organization is challenging because every organization is different. This paper will look at how organizations can approach developing an enterprise data architecture and use model-based systems engineering to support the design, development, and decision-making. An example of using the decision framework is described in Section 3.3, using a hypothetical organization so that other's might see how the framework can be used and the value the decision framework could have on their development process.

1.1.1 Enabling Data Stakeholders

Stakeholders and Use Cases An enterprise data architecture needs to support several types of users and use cases [3]. The goal is to enable the functional stakeholders while meeting the nonfunctional stakeholder's requirements. Common functional operations are supporting business analytics, data analysis, dashboarding, application development, machine learning, and artificial intelligence [12]. For each of these use cases it takes a team of people with varying skill levels to support each of these efforts [13]. Data Scientists, data engineers, developers, database engineers, statisticians, analysts, operations researchers, etc. are the functional data stakeholders the data enterprise environment needs to support [14]. To enable each of the efforts the data enterprise architecture should also support the tools, processing requirements, and workflows the teams require [13].

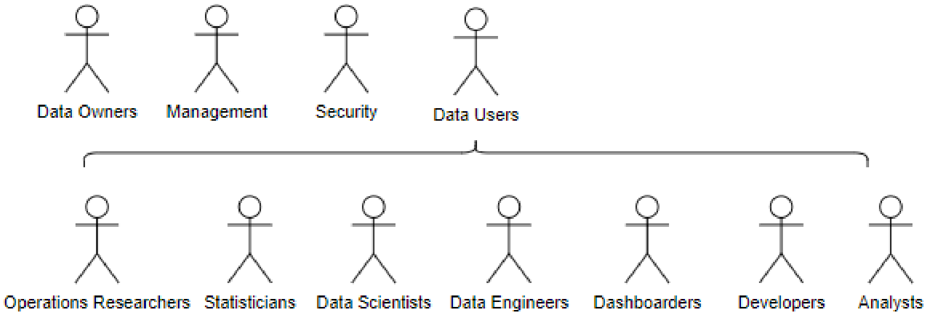


Figure 1.3: Stakeholders of an Enterprise Data Architecture

Here are some of the high-level scenarios the data architecture needs to support for the majority of the functional data users needs [3]:

- Users need to be able to discover data available for their projects and request access if they don't already have access.

- Users need to be able to understand where the data came from and how it was collected and modified.

- They need to be able to access the data in multiple ways. Examples: APIs, Queries, Downloads/exports via a GUI, Commercial Software etc.

- Users need to be able to request to add new data into the environment.

- The system should support batch and streaming data ingests.
- The system should support extract, transform, and load (ETL) workflows.

Non-functional stakeholders include data owners, security, and managers [15]. This group of stakeholders is important when meeting the needs of an organization. Their needs are what make the architecture unique to the organization, because they represent the risk tolerance, funding, and business operations.

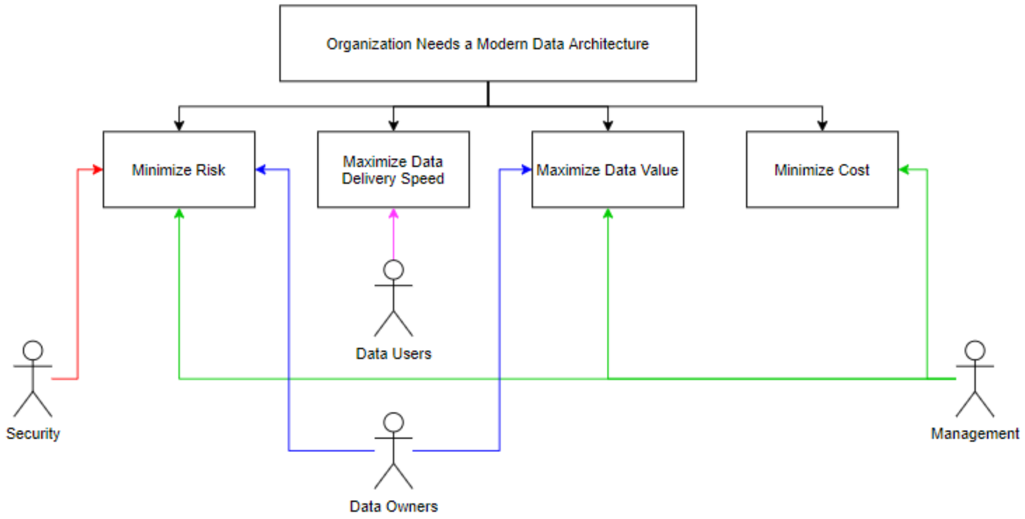


Figure 1.4: Objectives of the Stakeholders

Data Owners are the original stewards of the sources of data and usually own the responsibility for keeping the data safe [15]. They are the authority on making decisions for who can access the data and for what purpose. Data Owners are concerned with maintaining the quality and sensitivity of their data [17]. They need to be able to set access control permissions and define roles, groups, or attributes of who can have access to their data source [18]. They are concerned with who is accessing their data and for what purpose, this is important for satisfying data security and auditing requirements. Data Owners have conflicting desires of wanting to prevent data spills of their sensitive data from happening, but provide access to groups and teams that show the value of using their data set [18]. Not all Data Owners will have the same risk posture, which is important to understand when approaching a new source of data to bring into the architecture.

The security stakeholders are concerned with preventing data loss, identifying vulnerabilities, mitigating security risks, and preventing harm to the organization [8]. They are concerned with data security and preventing unauthorized access to data, same as the data owner stakeholders [18]. Security stakeholders are also concerned with preventing malicious actors from gaining access to the organizations systems and causing harm. They care about the resiliency of the system against external and internal threats [19]. They care about having an audit record so they can monitor the system effectively and have tractability in the event that a security breach occurs [18]. The enterprise data architecture should meet the organizations standards for security.

Management, program managers, or business owners are concerned with setting up their programs for success [14]. They are concerned about budgets, resources, timelines and achieving the goals of the organization. For a data science effort, they want to make sure their team has the resources they need including data and tools [19]. They support the data owners and security teams to make sure they keep the organization safe from preventable risks. Managers are also concerned with the organizations bottom-line and having to do resource allocation across the organization [20]. The enterprise data architecture is an investment they have to support. It needs to be within budget, maintainable, and deliver value to the organization.

Currently, organizations are likely to have architectures that meet some of the above stakeholder's needs. However the goal is for an enterprise data architecture to meet all of the stakeholders needs. Systems engineering can be used to map out what the gaps are in the current architecture and ensure the future enterprise data architecture addresses them. The development team's challenge is to balance all the needs of it's stakeholders [10]. Some of the end-goals maybe in conflict with each other and trade-offs will need to occur. Tough decision making needs to occur to deliver on each of the stakeholders requirements while balancing performance, schedule, and cost.

1.1.2 Current Architecture Problems

Many organizations current architectures are not set up to support data science efforts which leads to excess waste in time and money for an organization [20]. Data is hard to identify for data science projects because most organizations don't keep track of all their data assets. Data is also hard for users to get repeatable access to because the original data collection systems weren't designed for the kind of user access data stakeholders require now in order to perform advanced analytics [17]. Data security is another impediment to enabling data science efforts because many current systems don't have permissions defined for access to the raw data. A data science project will then either get some of the data or no data at all. Data science projects can still be accomplished, but at great cost and over a longer period of time than necessary [17]. A lot of time is wasted looking for the required data and trying to get access to the data in order to begin the next stage in the modeling process. Much of the modeling work is extracting, transforming, and loading (ETL) the data into a format that can be modeled. Most current architectures do not have a process or provide a place to store cleaned or modified data. This means data science efforts are continually performing the same ETL process for the same data sources which is another waste of time and resources [20]. An enterprise data architecture is aimed at reducing this overhead work by cataloging data, automating access, defining data permissions, and providing a central location to store cleaned data which reduces duplication of effort.

Organization's current architectures also require a lot of overhead maintenance costs [19]. Every separate data system might need a different development team and operations team to maintain it. The complexity of different systems with different designs means it takes more time for developers and security engineers to understand the data and architecture which makes system integration more challenging. Organizations also incur more risk when they have multiple systems to maintain. Monitoring multiple systems for unusual activity and making sure each system is compliant with security policies takes additional time and effort [19]. The risk of data loss or theft is higher because those threats can more easily go undetected. Current systems might not be designed to support the demand of data analytics users or DDOS attacks leaving the systems vulnerable to being taken down [16]. An enterprise data architecture can improve the standardization security, architecture design, documentation, scaleability, and resiliency which will reduce the risks and overhead maintenance costs [19].

1.1.3 Enterprise Data Architecture

Enterprise Data Architecture Overview An enterprise data architecture is a data management environment which is designed to support an organization with a large volume and variety of data [12]. Organizations have over time grown their data holdings by developing applications that collect data or purchasing data from other collectors [14]. Having multiple data sources helps to fill gaps in information for organizations, but they run into the issue of where to integrate, store, and process the data.

Organizations end up with multiple data infrastructure systems to maintain which causes problems for the users, data system owners, the security team, and the finance team. Data systems designed and managed separately are called data stove-pipes [11]. These stove-pipes lead to several issues within an organization. Data owners become protective of their data which can make it hard for users to get access to the data they need. Every system can have a different data owner and access procedure which can cause major delays in a data dependant project. The data system may not have been designed to scale to support advanced analytics [17]. The system might not have a security setup that allows for data sharing or it might not have the processing capacity to handle enterprise use. Systems developed in isolation from each other makes data integration between data sources challenging. Data integration ends up happening elsewhere and can be hard to security to track, which can lead to several security risks and data quality issues. Sometimes organizations don't even realize what data they have which can lead to duplicate efforts and risks of data spills.

An enterprise data architecture helps to address these issues. It helps to catalog all data assets and makes them discoverable to users [19]. It automates data access so users have a consistent process to follow and data owners have better knowledge of who has access to their data [19]. By centralizing data management the organization's security can keep better track of modifications to the data and detect any suspicious activity. An enterprise data architecture should also address the scalability issues and streamline the maintenance of their data systems [19].

There are a lot of requirements an enterprise data architecture needs to support [3]. There are a lot of options for building an enterprise data architecture to meet those requirements. This can make it challenging to decide which architecture is the best option for an organization to enable everything they want the system to do [21].

1.2 Problem Description

EDA Design Problems The goal of creating a new data architecture is to improve enabling data-driven efforts, reducing risks, and reducing overhead costs. However there are many problems and challenges when designing and building an enterprise data architecture because they are complex systems [10]. They have a lot of users, use cases, and options for implementation [3]. No two organizations have the same data, IT infrastructure, and risk posture which creates additional challenges in designing a system [8]. There are several components of an enterprise data architecture that are necessary for all designs in order to meet the functional and non-functional requirements, but their implementations can vary widely [22]. Every organization will have some level of customization to their EDAs. Organizations need to determine what components to build or buy [23]. Organizations need to determine how to implement their data security. Organizations need to determine what external systems their EDA should interface with. These decisions are unique to each organization and should be decided based on their current architecture, risk posture, requirements, and objectives [3]. Many organizations struggle to successfully design and implement an enterprise

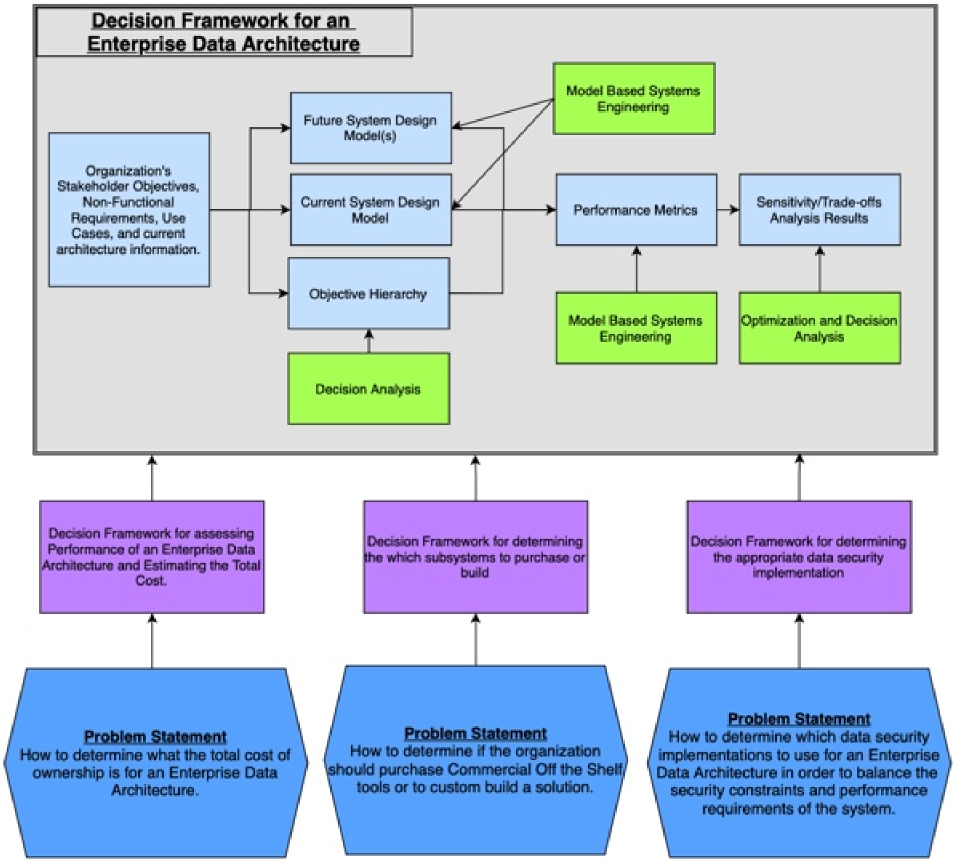

Decision Framework for an Enterprise Data Architecture
Organization's Stakeholder Objectives, Non-Functional Requirements, Use Cases, and current architecture information.
Future System Design Model(s)
Current System Design Model
Objective Hierarchy
Decision Analysis
Model Based Systems Engineering
Performance Metrics
Sensitivity/Trade-offs Analysis Results
Model Based Systems Engineering
Optimization and Decision Analysis
Decision Framework for assessing Performance of an Enterprise Data Architecture and Estimating the Total Cost.
Decision Framework for determining the which subsystems to purchase or build
Decision Framework for determining the appropriate data security implementation
Problem Statement
How to determine what the total cost of ownership is for an Enterprise Data Architecture.
Problem Statement
How to determine if the organization should purchase Commercial Off the Shelf tools or to custom build a solution.
Problem Statement
How to determine which data security implementations to use for an Enterprise Data Architecture in order to balance the security constraints and performance requirements of the system.

Figure 1.5: Framework Applications for Different Problems

data architecture [14]. They are an expensive project for an organization and it is critical they are successfully built. They are a strategic high-risk and high reward investment.

Estimating Total Cost of Ownership Problem The total cost of ownership for an organization's current and future architectures are important data points for decision makers to know before they can approve a modernization project for their data architecture [24]. Comparing the old architecture costs to the new architecture's cost by gathering information and putting together the development costs, the transition costs, and the operations and maintenance costs of the new enterprise data architecture is an important data point for the decision maker to have [24].

Deciding to Buy or Build There are many options for implementing components of an enterprise data architecture [25]. Many commercial companies have developed solutions which may be the right choice for an organization [22]. However there are risks with every decision that should be analyzed [24]. The industry is evolving rapidly and an organization will want to make sure if they purchase a solution it is capable of keeping up with that change. Some commercial solutions could be limiting the organizations ability to adopt new technologies as they become available [24]. Commercial companies can also change their pricing model or decide to no longer support or develop a capability further. These are risks that should be weighed when making a decision to purchase a solution instead of building one [24].

Building a solution is expensive [26]. Building a solution means a company needs to hire and maintain staff that can continually develop and operate the system which a long term cost to consider [14]. The benefit is developing an in-house solution means the architecture can be customized to the needs of the organization and simplifies the project resourcing by not having to manage additional contracts or licenses. Deciding whether to develop or build a solution is an important decision with long term planning [24]. An organization would benefit from using a framework to understand the decision trade-off in cost and schedule.

Deciding on Security Implementations Data security implementations can have a significant impact on the performance of the architecture [15]. A system architect will want to know how each of the security implementations effect the performance of different areas of the architecture. How the different designs can effect cost, data storage, processing time, or if they require a higher level of maintenance [26]. They will want to understand what security benefit is gained from each of these implementations and potential downsides. Depending on the risk posture of an organization, architects should know what would the best security implementation be and would the resulting cost be worth it? They should be able to inform the organizations decision makers on how their risk posture will effect the development schedule, development cost, maintenance cost, and performance of the system [26].

1.3 Scope

Problem Scope Overview Development teams are likely to encounter difficulty making decisions on an architecture because there are many stakeholders who have competing objectives [27]. This means development teams struggle to identify requirements and scope the system to a final architecture design. Current development methods do not provide a robust decision framework for determining an enterprise data architecture [14]. Enterprise data architecture projects are especially vulnerable to scope creep, solution bias, and scrutiny by an organization's leadership. There are many stakeholders influencing the decision making and many implementation options to choose from [28]. An organization's leadership will likely be more involved in the development of this project because EDA's are expensive investments and strategically important.

Development process methods do not provide development teams with the data needed to justify the development team's decisions on an architecture [27]. After the development team has selected an architecture and are in the build phase of the software development life-cycle, their architecture choice may be questioned. There are several scenarios a development team will need to answer to and justify their decision to the organization's decision makers [29]. Depending on the development process chosen the development team may or may not have the information needed to quickly respond and will need to stop work on the project in order to research and answer those questions.

Changes in the organization's environment can have significant impact on an enterprise data architecture project [30]. New leadership will want information on how an architecture was chosen. They might have new requirements or are aware of an architecture solution the development team wasn't aware of. The development team will need to be able to answer questions and justify their architecture decisions [29].

1.4 Research Questions and Hypothesis

Research Questions To improve the success of an enterprise data architecture being developed for an organization the development teams process must become more robust with decision support information. To determine how to improve the process the following questions were researched to identify improvement areas.

1. What development methods are currently used to develop enterprise data architectures?
2. What information gathering and analysis do current development methods provide?
3. What information gathering and analysis do current development methods not provide?
4. What can be used to improve the decision making and justification of development teams?

Question #1 was researched to understand current development processes and how they are used to build an enterprise data architecture. Question #2 was researched to understand how those processes support teams making decisions on their architecture. Question #3 is to understand the limitations of the current development processes and why development teams currently struggle. Question #4 is looking for other methods that can address the gaps current development processes have for supporting decision making.

Hypothesis To improve the decision making of the development team and reduce the impact of external influences on the project, the development process should be supplemented with a value and data driven decision framework. It will help identify the best architecture approach and support justification of that decision throughout the development life-cycle. The hypothesis is the following statement:

"The development process for an enterprise data architecture can be improved by enabling value-based data driven decision making on the architecture design and reducing leadership question response time throughout the development life-cycle."

If teams have the information they need to determine the best architecture for their organization and have the data to back up that decision making, their project is more likely to be successfully built. Teams having the information they need to respond to leadership questions quickly will reduce the time devoted to reacting to external influences and allow the team to focus on building out the capability.

Chapter 2: Literature Review

Overview This chapter will provide the literature review and research relevant to the research questions. Beginning with a discussion of why big data projects fail and the challenge scenarios development teams encounter. It will cover the current research that's been done on enterprise data architectures and the development processes that are commonly used to build them. It will go over how those processes work and the information the development team collects while working through the process. There is a discussion on some research that has been done on improving agile development processes to support the development of big data architectures. There is a section on the research on how software development teams make decisions. How organizations effect decision making of development teams and whether documenting decisions is useful or not.

The final section is on with two alternative methodologies that can be used to support development teams decision making on enterprise data architecture designs. The Decision Analysis methods for supporting obtaining the primary objectives of the system and evaluation of alternative designs. Model-Based systems engineering methods for documenting the enterprise data architecture designs.

2.0.1 Big Data Architecture Project Failures

A survey from 2019-2020 collected data from 89 big data projects on their success rates [2]. The results of the survey were compared to another survey conducted in 2018, that looked at success rates of IT Projects in general [2]. The 2019 survey found that big data projects had a failure rate of 67.42% and a success rate of 21.35% [2]. The remaining 11.24% of respondents said it was too early in the project to determine success or failure [2]. Of the big data projects that failed, 51.69% partially met the requirements for the project and

11.24% failed to meet their project requirements [2]. The results from the big data project survey found a dependence between the project being considered successful and whether it met the requirements and delivered on time [2].

The 2018 survey, found the overall success rate of IT projects based on responses from 149 respondents was 52% successful, 40% challenged, and 8% failed [31]. Based on the results from both these surveys it's evident that big data projects struggle more than other development projects [2]. Big data projects are more likely to completely fail or only partially meet the stakeholder's requirements [2].

Comparison between the two surveys also found that big data projects are generally considered more complex than other IT projects [2]. For new development 65.75% of big data projects were considered high risk and complex compared to 10.59% of other IT projects [2]. The author interviewed a respondent to understand what makes big data projects complex [2]. Their responses implied that getting quality data was important, understanding the context of where the data comes from was important, and having the people with the right skill-sets to work with data were key drivers in the success of a big data project [2].

Challenge Scenarios There are common situations development teams run into where they need to adjust their requirements and development plan [29]. This is why agile has become so popular because one of it's core principles is to embrace change in requirements and allows the teams to quickly pivot and adapt to the changes [32]. Depesa, listed six common scenarios based on data collected from LinkedIn [29]:

1. The project owner identifies new business opportunities and decides to integrate them into the software being developed.

2. Due to the technical nature of software development projects there is a lack of shared understanding of expected outcomes.

3. Original planning was based on specifications that were misinterpreted by the project manager or poorly illustrated by the project owner.

4. Project team is unable to implement planned functionalities due to lack of expertise or technological limitations.

5. The context in which the software is going to be used changes thus generating the need for the software to change

6. New technology or software product is launched on the market.

The negative impact of these scenarios is discussed in the paper. Frequency of changing the projects specifications leads to: Jeopardizing deadlines, going over the projects budget, and causing stress and discontentment for the development team [29]. Keeping up with the dynamics of technology and standards leads to: software becoming obsolete and the development team needing to invest a significant amount of time researching new technologies [29]. A skilled workforce had the positive benefit of increasing the likelihood that the project would be successful, however it increases the cost of the project [29].

2.1 Development Processes

Big Data Architecture Development Research Research on development processes for big data architectures is an emerging field of research. Laigner et al, conducted a mapping study of the research on software engineering approaches to developing big data systems [33]. The study identified only 52 papers published from 2011 to 2016 that answered the primary research question "Which types of software engineering approaches have been proposed to support developing big data systems." The study found most research is focused on development methodologies and software architecture [33]. The development methodologies papers were looking at how to modify development processes to better address the complex challenges of big data systems. The software architecture studies were looking proposing different architecture design alternatives.

After reviewing the studies the paper referenced, there were only three papers that focused on supporting the decision making of an enterprise data architecture. None of the papers proposed a method for evaluating different architecture alternatives, they only proposed different methods for capturing requirements and developing the architecture.

Although there are not many studies comparing the different development processes for building enterprise data architectures or big data architectures, there are many studies comparing development processes for other software development efforts. There are a lot of different development processes and models that have been developed over the years. The most common ones in use are agile, waterfall, v-model, and hybrids of the development processes [30].

Most of these are researched from the perspective of fully developing the capability from scratch. Another thing to consider is if there are already solutions available, whether to modify an existing system to meet the requirements or buy a solution [34]. There are many factors for organizations to consider when making a decision on what their enterprise data architecture should be.

2.1.1 Common Development Methodologies

Vijayasarathy et al, conducted a survey of development projects and asked what methodology they used [35]. The survey also asked questions about the organization and what type of project was being developed [35]. They had 153 respondents to the survey. The most common development process was Waterfall, 32%, followed by Agile Unified 28.1%, and Scrum 20.3%. But many of the project surveyed used multiple development methodologies, so the authors grouped the development processes into the following categories: Hybrid 45.2%, Agile 33.1%, Traditional 13.8%, and Iterative 7.7%. Hybrid was the most common approach for development projects.

They found that 55.5% of projects that used traditional development methods were “high-revenue companies ($> \$1B$)” and had more than 10,000 employees [35]. Agile was used more often by companies with less number of employees and a medium sized revenue [35].

The survey also asked about how critical the system was, and the authors grouped the responses into low, medium, and high criticality. A significant majority, 88.2%, of development projects that used traditional development methods said the projects were high criticality systems [35]. Where 51.2% of agile projects said they were high criticality [35]. The authors stated, “This data suggests that organizations tend to use traditional approaches on critical projects” [35].

The results of this paper’s survey were compared to their previous survey in 2003 [35]. They found that agile had grown in popularity and was likely to continue this trend [35]. And based on this trend, development teams are likely to use agile or a hybrid of agile to develop an enterprise data architecture [35]. However given the data from this survey, an enterprise data architecture is likely to be built using traditional methods as well, due to it being a critical system for large organizations.

Development projects were 38.6% new software development project. However asked about other types of development projects. Software enhancement was 24.2% which was the second most popular option. Customizing commercial off the shelf was the third most popular option at 13.1% [35]. Based on this data the most common approach to a new project is to build, instead of modify existing projects or buying pre-built solutions. However this includes project of all sizes. Looking at large complex systems might show a different pattern. The author didn’t provide the data view from a large project perspective. Scott Wambler conducted a survey of development projects in 2018 and found comparable results [31]. The findings from the survey were that there were so few traditional responses compared to other agile methods that they lumped the traditional responses with adhoc types of development methods for their analysis [31]. This shows a general trend of agile becoming more popular for general development projects [31].

Traditional Development Processes The Vijayasarathy et al, grouped several development processes under the traditional umbrella term [35]. These methods are characterized by being more plan driven and sequential [35]. Other researchers have grouped methodologies into “heavyweight” and “lightweight” methodologies [29]. Heavyweight similar to the traditional term, characterizes these methods as easy to understand, emphasize the use of documentation, and are appealing to management because of their structured approach which makes them easier to resource and track [29] [30].

Waterfall In 1970 Dr. Winston Royce proposed the waterfall development method [36]. It outlined the seven key phases a software project needs to go through to develop an application. The seven occur in a liner approach, where one phase cannot begin without the current one being complete. Dr. Winston highlighted that if problems are identified in later phases of the development, the risk that the development project may need to start back on the first phase and rework the whole process over again.

The characteristics of the waterfall process is it provides comprehensive documentation, meticulous planning, it's a linear sequential process, and each phase has it's own deliverables [29][30]. The benefits are that it's an easy process to manage and it's easy for the product owner and development team to understand [29][30].

Dr. Royce emphasised the importance of documentation in his 1970 paper. He outlined many reasons why it is important to have documentation complete at every phase of the development process. First it improves communication between designers, management, and stakeholders of the system. It improves testing of the system and maintenance of the system.

However the waterfall process does not adapt well to changing requirements or errors found later in the process [29][30]. It also delivers working code later in the development cycle [29][30]. The amount of documentation slows down the speed of development [37]. Although Dr. Royce's original paper was titled for large application development, recent studies have classified it as a process that should be used for only small projects that have their requirements clearly specified [30].

V-Model The V-Model is an extension of the waterfall sequential process with the added emphasis of developing test plans at the end of each phase [30]. The testing validates the requirements, design, and software [30].

The characteristics of the V-Model are that it introduces testing at every phase in the development process and is focused on maintaining the system [29][30]. The benefits are it has a low bug rate, less error prone, and it's easy to follow. The weaknesses are it is vulnerable to scope creep and relies on the initial specifications heavily, which means it's not very adaptable to change [29][30]. Recent studies have classified the V-Model as appropriate for small and medium sized development efforts [30].

Agile Agile began with the publication of the Manifesto for Agile and it's twelve principles for software development [38] [32]. In contrast to the traditional development processes the manifesto states "Working software over comprehensive documentation" and "Responding to change over following a plan" [38]. The shift in paradigm has brought about a more iterative development processes that stress delivery of working software earlier in the process, which stakeholders can then test and provide feedback on [30].

The Scrum method is characterized by iterative development, time-boxed cycles called sprints, short daily scrum meetings, self-organizing development teams, and task management using backlogs [29] [30]. The benefits are quick delivery of working software, fast feedback, rapid adaption to change [29] [30]. The weaknesses are lack of documentation, requires more experienced developers, more difficult to track and predict delivery dates and cost [29] [30].

Acquisition Process Outsourcing, buying whole solutions or individual components of an enterprise data architecture is another process option that should be considered by organizations. The decision to buy a commercial capability is non-trivial [39]. There are many vendors that offer similar solutions for the same capability, for example in 2015 there were 150 products for a NoSQL database [39]. New technologies come on the market frequently and replace old competitors, which can make it difficult to pick a relevant and sustainable choice [39].

Advantages to purchasing an "Off-The-Shelf" capability are the software is likely to have less bugs, the software likely will come with training and support, it's usually more cost effective, and the software is likely to continue on improving over the years [40]. However the software might need to be customized in order to fit the needs of the organization, because it either doesn't meet all the requirements or it has some integration conflicts with other software the organization has. This customization will increase cost and the delivery timeline [40].

One paper concluded that there were three influencing factors in the decision to buy or build a large-scale application [34]. The transfer of development risks, the long term maintenance of the software, and the strategic importance of the software [34]. An enterprise data architecture might want to consider this solution, but be aware of the integration and modification risks.

2.1.2 Improved Processes for Big Data Architectures

Two studies that the Laigner mapping study found offered modified development approaches to support building Big Data Architectures [33]. Both papers are by the same authors and build upon their solution for improving the agile development process to support developing big data architectures.

The first paper proposed a modified agile approach that tries to find a balance between over-architecting and no architecture design work. They supplement the agile development process by adding a system architect to the team to help document and analyze the architecture that's being built [41]. The development process begins with a value discovery phase where the system architect, development team, and stakeholders work to identify the business goals of the system, the functional requirements and system constraints [41]. The architecture design is captured using the Big Data Design (BDD) Method which is a big data version of the Attribute Driven Design (ADD) method [41] [42]. The BDD method uses common system engineering diagrams to describe the system such as data flow diagrams [41]. One key step in the BDD method is selecting a reference architecture to base the design on [41]. The team iteratively begins to select and test options for components of the system based on the reference architecture [41]. They capture decisions made every sprint cycle, but do not mentioned how those are recorded. After a design iteration is complete they evaluate the design for risks and design trade-offs [41].

This improved process does a good job of documenting only the essential needs so the development team can be focused on developing. This process takes the best of both the agile development principles and traditional documentation based development processes to support building big data architectures. Aligning the system to the organization's values is another good idea this process identifies as a key step.

This modified process was proven to be successful for 10 case study projects [41]. However those projects were under contracts that had clear scope and strict budget and schedule constraints [41]. The authors also noted the organizations that were building the big data architectures were had a innovative and risk accepting culture [41]. This likely means the management for these projects provided less scrutiny than other less risk accepting organization's might [41].

The next paper adds Strategic Prototyping as part of the development process. This paper advocates developing targeted minimally viable products (MVP) to test out components or features of a big data architecture [43]. Usually this is to test non-functional requirements such as performance and scalability [43]. They argue "architecture analysis alone is insufficient to prove many important system properties" and this is why prototypes are needed to gather the information needed in order to make a final decision on a design [43].

What is lacking from both papers is an understanding of how the architecture decisions are made using their BDD method and how the design trade-offs occur. The paper's do not go into detail on how technologies are chosen based on a reference architecture. The process relies on quick decision making and testing, which can be good for more risk accepting organizations. However some organizations are risk intolerant and don't have the flexibility for prototyping. For example an organization might not have the acquisition flexibility to try one commercial solution for a component and throw it away if it doesn't work. If they are going to go through the process of acquiring a solution it better work.

The ADD 3.0 document, which is the version the BDD process was referencing, walks through how the attribute driven design process works [44]. They argue that quality attributes are what drive the architecture decisions and those are: performance, modifyability, testability etc [44]. The quality attributes need to be prioritized by the stakeholders of the system [44]. But there isn't a discussion of how the design trade-offs are analyzed. The process has a step about determining a reference architecture and selecting technologies that map to the components of that reference architecture but it doesn't go into detail on analyzing which technologies might best fit that reference architecture.

The ADD 2.0 document outlines the general design process for software architectures [42]. It outlines the steps for gathering and prioritizing requirements, decomposing the system into component elements, and categorizing the requirements based on stakeholder impact and technical impact [42]. It then goes into how to brainstorm different design patterns and develop a matrix that lists the pros and cons of each design pattern [42]. It covers the details an architect should consider when making design decisions [42]. The paper emphasises that the design process is iterative and that architects should document decisions and assumptions [42]. However it only references the trade off analysis technique called Architecture Tradeoff Analysis Method (ATAM). The method has been used for 15 years to evaluate software architectures based on the quality attributes [45].

The analysis method builds objective models to calculate how an architecture would perform against an attribute requirement. The paper goes over how to model availability, performance, and security of an IT system [46]. Each model is built separately and the paper demonstrates how some objectives can have positive correlations and others negative [46]. It shows how an architecture design can be measured. However the paper does not show how to balance the tradeoff between the negative and positively correlated attributes. The architect has values for how each architecture measures against the different attributes, but not a comprehensive value ranking for each of the architectures.

2.2 Software Architecture Decision Making

Overview In 2019 Razavian et al, published did a review of the empirical research on software architecture decision making [5]. They broke up and categorized papers into two groups: Decision Making Practice and Decision Making Behavior. Decision Making Practice includes research on decision processes, techniques, and tools to support software decision making, which are the primary topics of interest for this thesis [47]. The study found 25 papers between 2005 and 2017 that are on the topic of decision making practices [5].

Decision Making Behavior research is focused on human decision making behavior [47]. The empirical research only found 13 papers focused on understanding the behavior and bias behind humans making decisions on software [5].

A paper called "Decision Making in software architecture" is referenced by the empirical research, but not included in the final 13 papers [5] [48]. It has a good overview of the decision making behavior topic and the fundamental issues that influence software architecture design [48]. It discusses the different bias's that can occur in software decision making including: Anchoring bias, Framing bias, Confirmation bias, Group-think bias, and other cognitive biases [48]. It discusses the difference between rational thinking and intuitive thinking [48] [49]. It talks about the Naturalistic decision making versus Rational decision making. Natural decision making are made with unconscious emotions and subjective domain contexts [48]. Rational decision making quantitatively compares options [48] [50]. Enterprise data architectures will need to have both behavioral types of decision making occurring to enable the unbounded innovative thinking to develop alternatives, but the objective decision assessment of the alternative architecture designs.

None of the final papers in either category contained research specific to decision making for big data architectures or enterprise data architectures.

Influences on Decision Making Groher and Weinreich a wrote a paper on the organizational influences that impact decision making on software architectures [28]. They interviewed software architects, team leads, and senior developers to understand what impacted their decision making [28]. They found that 96% of the respondents were influenced by the organizational structures and processes [28]. Group decision making on architecture was common, 68% of the people interviewed reported the development team decides together. The other 32% said there were dedicated roles for making architecture decisions. They found that 80% of the participants used an agile development process. The other 20% used a traditional plan-driven development process.

One interesting anecdotal statement from a respondent was, "revealing too much about a decision can make trouble because you have to spend a lot of time justifying your decision. A solution is often to simply present a plausible design and discourage the discussion of alternatives" [28]. The author's interpretation was "a defined decision-making process in place, which requires the explicit documentation of decisions, their rationale, and alternatives, may actually be a hindrance for the consideration of different alternatives and thus may discourage an open reasoning process" [28].

An alternate interpretation could be that being transparent about the decision making and how the team made a decision on their architecture is important. The issue seems to be the struggle in justifying the decision making. This could be because the decision making methodology or information was not good. Discouraging discussion of alternatives means limiting options and potential better solutions.

Group Decision Making Rekha and Muccini conducted a survey to gather information on group decision making on software architectures [51]. They had 35 respondents to their survey. The majority of respondents, 46% worked for large organizations, more than 999 employees [51]. Small companies were 31% with less than 100 employees and medium size companies were 23%, with between 100 and 999 employees [51]. From the survey the authors had several important take always. They found that for group decision making documentation was "common practice, with goals and requirements almost always documented". There were multiple responses about the lack of tools to support group software architecture decision making. Many decisions were under time constraints and were not able to complete a full assessment of alternatives. The author suggests this could lead to issues like group-think. The authors propose blending structured and unstructured approaches to group decision making, where structure is added wherever required.

Agile Decision Making Drury-Grogan et al, wrote a paper that outlines how agile teams make decisions [52]. The paper explains how value-based decision making occurs using the four core agile values and how that impacts agile team's decision making. They made several conclusions based on data collected from a case study. There were a few take always that could negatively effect the decision making of an enterprise data architecture if the agile process is used, following the four manifesto values.

1. **Value individuals and interactions over processes** means "repetition of decisions during and across iterations and domination of more experienced staff in decision making activities" [52]. Many decisions were repeated because there was little discussion about trying something different, which mean teams would encounter the same problems.

2. **Value working software over documentation** means "decisions are made on poor information intelligence" [52]. The author said "Because the team aimed to minimise documentation, decisions were made using ad hoc, inaccurate, incomplete, or non-existing documentation" [52]. The teams lost data and forgot decisions because there was no centralized documentation. Decisions were made on multiple communication channels or whiteboards and not easily communicated or available for reference. "Decision intelligence was challenges as no support system exists to automatically present all data with a single, clear, accurate view to co-located and non-co-located team members" [52].

3. **Value customer collaboration over contract negotiation** means decisions depended heavily on the customer representative for the project and their technical expertise and experience [52].

4. **Value adapt to change over following a plan** means discussion of complex issues was deferred and short term decisions were prioritized [52].

Is Documentation of Decision Rational Useful? In 2005, a study was done to determine if documentation of architecture design rational was useful [53]. They conducted a survey and received 81 responses from architects and designers. The survey asked about how important generic rationals were to their architecture decision making. But several of the respondents added additional rationals they used to make architectural design choices and the researchers grouped them into three broad categories:

1. **Business Goals** such as enterprise strategies and adherence to industry standards.
2. **Requirements** which included functional, non-functional, and whether to build or buy.
3. **Constraints and Concerns** which included compatibility with existing systems, current IT architecture and capabilities, viability of solutions, and time availability.

These concerns are still relevant and especially to determining enterprise data architectures. The paper concluded that having decision rational is an important part of the design process, that it should be documented, and that it was helpful in justifying their design choices [53]. The paper did highlight some issues preventing teams from documenting their decision rational. The three big ones were not having tools to support documentation, not having standards or guidelines of how to do it, and not having the funding or time to do it.

2.3 Alternative Methods

All types of development processes are used to build an enterprise data architecture. But not all development processes approach the architecture design phase or documentation the same way. Depending on the organization, different development processes are used. The organization also influences how decisions made by development teams. Tools to support documentation and decision making are not commonly found.

To improve development teams decision making on enterprise data architectures, they need a method for supporting their decision making and documenting their architectures and decisions. The following section provides research on two techniques that can be used to support development teams.

2.3.1 Decision Analysis

Value-Focused Thinking Ralph Keeney wrote the preliminary paper on value focused decision making [54]. He offered a new paradigm of making decisions based on the values of a decision maker in the context of a decision problem [54]. This was in contrast to the standard "alternative-focused thinking" approach where decision makers are focused on identifying alternative solutions first then identifying criteria for evaluation. He made the argument that "it is the values that are fundamentally important in any decision situation, more fundamental than alternative, and they should be the driving force for our decision making".

The key difference in approach is that value focused thinking emphasises thinking hard about the problem, by understanding what the objectives are first, then identifying criteria for evaluating alternatives against those objectives, and then lastly coming up with alternatives to evaluate. Keeney advocated that this method would allow for the generation of better alternatives and help uncover hidden objectives decision makers may have [54].

Multiple Objective Decision Analysis Value focused thinking can be used to capture the intent of a decision maker and guide strategic thinking [55]. Value-Focused Thinking uses Multiple Objective Decision Analysis (MODA) as a mathematical technique to implement the Value Focused Thinking philosophy [56]. Multiple objective decision analysis is an operations research technique used to determine the best alternative when there are multiple conflicting objectives [56].

Parnell, outlined how this method can be used to support several complex military applications, which include several IT systems [56].

Definitions Parnell outlined in his paper [56]:

1. **Fundamental Objective**: The most basic objective looking to be achieved.

2. **Objective**: A preference statement about an evaluation consideration. To maximize or minimize.

3. **Value Measure**: A scale to asses the degree to which an objective is achieved. Alternative terms are evaluation measures, measures of effectiveness, measures of merit, and metrics.

4. **Qualitative Value Model**: The complete description of our qualitative values, including the fundamental objective, functions (if used), objectives, and value measures.

5. **Value Hierarchy (Value Tree)**: Pictorial representation of the qualitative value model.

6. **Value Function**: A function that assigns value to a value measure score.

7. **Quantitative Value Model**: The value functions, weights, and mathematical equation (e.g., the additive value model) to evaluate the alternatives.

8. **Weights**: The weight assigned a value measure depends on the range of the value measure. Weights are our relative preference for value measures. Weights must sum to one.

9. **Value Model**: Both the qualitative and quantitative values models.

This paper outlines some software that can be used to support the multiple objective analysis, called Logical Decisions. However this method can be done using spreadsheet tools or by hand if necessary.

Strategic Decision Making Craig Kirkwood wrote textbook outlining a multi-objective value analysis approach to making decisions [57]. The book cites many case study's where the approach was used in multiple fields. The approach is very useful to decisions where “there are multiple competing objectives that require consideration of trade-offs among these objectives” [57].

The book proposes a quantitative approach to decision making. “This improves decision making, and it also aids communication about the basis for a decision. It is typical in modern business and other organizational decision making that a variety of stakeholders need to understand and help implement a decision” [57].

At a high-level the book outlines a strategic approach to decision making includes the following with the following five steps [57]:

- Specify objectives and scales for measuring achievement with respect to these objectives.
- Develop alternatives that potentially might achieve the objectives.
- Determine how well each alternative achieves each objective.
- Consider tradeoffs among the objectives.
- Select the alternative that, on balance, best achieves the objectives, taking into account uncertainties.

The method first identifies objectives and structures then in a value hierarchy. Measures are then identified for each of the objectives and evaluated using single-dimensional value functions. A value function is used to evaluate the alternatives against the objectives.

“To conduct a multi-objective value analysis, it is necessary to determine a value function, which combines the multiple evaluation measures into a single measure of the overall value of each evaluation alternative. The form of this function that is used here is a weighted sum of functions over each individual evaluation measure” [57]. The value function requires a “Single dimensional value functions be specified for each evaluation measure. (Single dimensional value functions are also called single attribute value functions.)” and “Weights be specified for each single dimensional value function. (Weights are also called scaling constants or swing weights.)” [57].

This method allows for alternatives to be ranked and compared based on the values of the decision. An alternative with that meets all the values perfectly, would have an overall value of 1. An alternative that fails to meet any of the values would have an overall value of zero. Alternatives that meet some values and don't meet others will have an overall value in-between 1 and 0. "The value number for a particular alternative gives the proportion of the distance, in a value sense, that the alternative is from the (possibly hypothetical) alternative with an overall value of zero to the (also possibly hypothetical) alternative with an overall value of 1" [57].

Using this method gaps can be assessed by comparing the value of the best alternative to the best hypothetical option [55]. This can help drive where improvements should be made in order to address the gap in value.

2.3.2 Model-Based Systems Engineering

Model-based systems engineering is "formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [58]. This is in contrast to the more traditional document based approach where information about the project is captured in several textual documents. There are different modeling methods under MBSE such as the Object-Oriented Systems Engineering Method (OOSEM) and the Functions-Based Systems Engineering Method (FBSE) [58].

The object-oriented method supports specification, analysis, design, verification of system, integration with object-oriented software systems, and reuse of the system design [58]. The core tenets of this method are requirements analysis, trade studies, and integrated product and process development.

Blocks are used to describe elements of the system similar to how UML defines classes. Common Diagrams used to describe the system are: block definition diagrams, parametric diagrams, use case diagrams, requirements matrix, sequence diagrams, activity diagrams, state-machine diagrams and more. SysML is the primary language used to create the diagrams and define the systems structure and behavior [58].

MBSE Benefits Henderson and Salado did a literature review to understand what the benefits of using model-based systems engineering (MBSE) are and what evidence there was for making the claim [9]. They found 360 papers that made claims about the benefits that MBSE provided. Although they found that there is no empirical evidence that proves the benefits of using MBSE, there are several informally observed benefits.

Henderson and Salado found two papers that claimed MBSE provided measurable benefits [9]. Increased traceability, improved consistency, reduced errors, better accessibility of information, high-level support for automation, and reduced burden of systems engineering tasks were all measured benefits cited in both papers. One paper also cited these benefits in addition to the ones previously mentioned: increased capacity for reuse, improved system quality, improved system design, better knowledge management capture, better requirements generation, higher level support for integration, improved architecture, increased rigor, and better requirements management.

More benefits listed in the papers which were observed but have not been verified by experimentation are: early V&V, increased efficiency, better analysis capability, improved system understanding, reduced time, and better communication & information sharing.

All these benefits would likely be very useful for complex projects like an enterprise data architecture.

MBSE Improving the Architecture Selection LaSorda demonstrated how MBSE could be used to support the architecture selection for three different case studies [59]. One was a System of Systems (SoS) communication satellite, where MBSE was used to help the acquisition process by demonstrating the feasibility of the system. Another was selecting an architecture for an electro-optical remote sensing satellite system and then comparing it to to the current Department of Defense process. The third case study was on the architecture selection for a service oriented architecture.

The paper demonstrates how MBSE can be applied to designing and evaluating different complex architectures. The evaluation and selection of architectures supported the acquisition process for these systems. The findings were that it provided better value, greater visibility into the decision making process and improved trust in the final decision [59].

MBSE Improving Software Development Patil and Annamaneni, wrote a paper explaining how MBSE can be used to support the development process [60]. Their paper comes from the perspective of automotive product design and development [60]. Their paper show's how MBSE is used to support every stage of the development cycle. The paper follows the V-Model process. They emphasis how useful MBSE is for making sure their system is compliant with standards and guidelines. Since they are designing software for vehicles, safety testing are extremely important to their development process. They want to make sure their design meets all their safety requirements.

Another benefit this paper emphasises is their models are reusable, which speeds up the development process in future iterations. The models are able to integrate with other tools commonly used to help manage requirements or test components of the system. [60]

MBSE Improving Decision Making Gebreegziabher et al, wrote a paper on how MBSE can be used to support decision making at every stage of the product development process [61]. The paper explains how MBSE diagrams are used to document the system at every stage of development. Requirements diagrams, use case diagrams, behavior and structure diagrams are used to describe the system. The diagrams show traceability between the requirements and how they are implemented in the system. All diagrams can be organized and described in a package diagram. Parametric diagrams are used to support trade-off analysis by describing how measures of effectiveness (MOEs) are calculated.
The paper stated MBSE provided the following benefits [61]:

- Provides a rigorous basis for technical decision-making.
- Resolution of requirement conflicts.
- Assessment of alternative physical solutions.

- Determine progress in satisfying system technical and derived technical requirements.
- Support risk management.
- Ensure that decisions are made only after evaluating the cost, schedule, performance, and risk.

Gebreegziabher et al's paper is another example of how MBSE techniques can be used to derive and document essential functions and structure of a system. The MBSE modeling language SysML is useful for decomposing and documenting details of a system so that it can be analyzed. The MBSE methods make it easier to communicate and document complex systems which would be beneficial to an enterprise data architecture development team.

UML vs SysML Systems Modeling Language (SysML) is the primary language used by model-based systems engineering. Unified Modeling Language (UML) has been the primary language used by developers and software architects to document software concepts. SysML is an evolution of the UML language meant to help with documenting systems with a lightweight profile [62]. Meaning it uses the diagrams and details necessary to document a system at a higher level [62]. UML 2.0 is still the preferred method for documenting details of a software such as data schemas, objects, and classes.

Two key diagrams that UML doesn't provide are parametric diagrams and requirements diagrams [63]. Parametric diagrams allow for analysis of a design and simulations. This is useful for estimating performance of an architecture design [63]. The requirements diagrams are useful as well for documenting requirements and mapping them to the the different components of the architecture. This helps show which components support which requirements. This also helps verification and validation of the system [63]. The diagram provides the ability to trace and track requirements to a component which was successfully completed and see which requirements still need to be full filled.

Chapter 3: Solution and Experiment

Chapter Overview This chapter will cover the details of the value model-based decision framework and an example of how it can be applied. The Value Model-Based Solution Framework Section 3.1 will explain each of the models, the data each model requires, the tools used to develop the model, the steps to creating the model, and expected outcome of each model.

The Experiment Design Section 3.2 will cover the background information for the example application of the framework. It will describe a hypothetical organization, their employees, and their goals for what they want an enterprise data architecture to achieve for their organization. The decision that this hypothetical organization ends up analyzing is whether to improve the current architecture or build the enterprise data architecture. But other organizations might center their decision making on whether or not to use a particular technology or whether to buy a commercial capability. The same approach can be taken to document the alternatives and come up with methods to analyze them based on the objectives of the organization.

The Experiment Models Section 3.3 will go into detail on how each model is developed for this organization. It will cover how the data would be collected, the tools used, and the resulting models. These models are combined to create a holistic picture of each alternative solution based on the organizations preferences. The value is calculated for every alternative and a final ranking of the alternatives is demonstrated.

The Framework Results Section 3.4 will compare the results of the current architecture alternatives to the results of the future architecture alternatives. The results are presented and discussed with the hypothetical decision maker. The decision maker provides feedback on the results. The models are then modified to incorporate the feedback and presented to decision maker again. This is meant to simulate real world discussions on enterprise data architecture designs and how decision makers could react to the information provided and how the models can easily be adjusted.

3.1 Value Model-Based Decision Framework Solution

Solution Overview This framework is a combination of multiple analytical and descriptive models which are used to evaluate data architectures against the objectives of an organization [64]. The result of the framework is a value that represents how well an architecture meets the objectives of an organization. The value result for each of the architectures can be compared to determine which architecture is best for the organization. The goal of the framework is to provide decision makers with the information they need in order for them to make the best decisions for their organizations. The framework gives them a tool to simulate the effects of their choices on an enterprise data architecture. It also helps to estimate potential costs of their decisions on the architecture design.

Each component model has a different process for development and requires different tools and data. Because some of the models require data from other models there is an order in which the models should be developed. Figure 3.1 is a diagram describing the order in which the models are created and the artifacts that come out of each step. The goal of each step is to achieve an executable model and updated documentation based on the information collected at that step. As the process continues the stakeholders and key decision makers should be consulted to make sure the models are accurate representations.

As data is collected throughout the process the models may be revisited. It is important to document changes and decisions as data is gathered throughout the process. Although there is a sequential flow of discovery and building these models, all models can be updated at any time. As new data and information become available or events occur or priorities shift decision makers will change their minds and this can be captured and measured with the models. When meeting with stakeholders information can be collected and saved for later models in the process. For example, when meeting with engineers to discuss the current architecture, future architecture questions can be asked as well. The phases outlined in 3.1

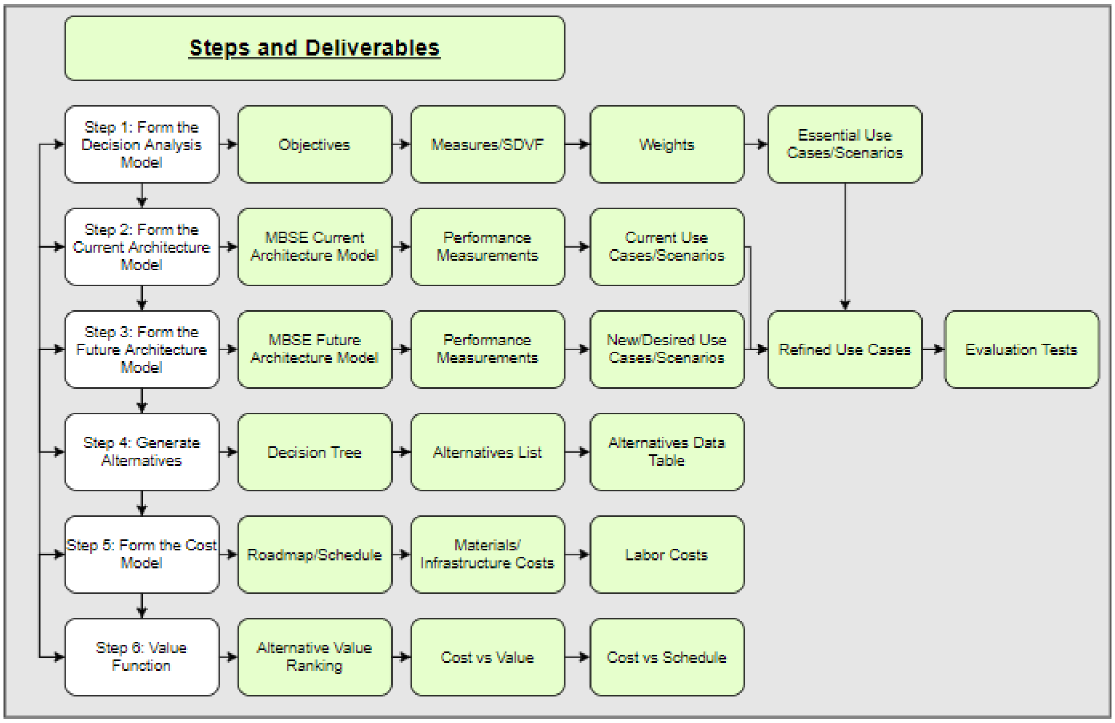

Steps and Deliverables
Step 1: Form the Decision Analysis Model
Objectives
Measures/SDVF
Weights
Essential Use Cases/Scenarios
Step 2: Form the Current Architecture Model
MBSE Current Architecture Model
Performance Measurements
Current Use Cases/Scenarios
Step 3: Form the Future Architecture Model
MBSE Future Architecture Model
Performance Measurements
New/Desired Use Cases/Scenarios
Refined Use Cases
Evaluation Tests
Step 4: Generate Alternatives
Decision Tree
Alternatives List
Alternatives Data Table
Step 5: Form the Cost Model
Roadmap/Schedule
Materials/ Infrastructure Costs
Labor Costs
Step 6: Value Function
Alternative Value Ranking
Cost vs Value
Cost vs Schedule

Figure 3.1: Steps for Developing the Value Model-Based Decision Framework

represent a sequential procedure with final deliverables as the goal to achieve at the end of each phase, but the actual process is much more iterative. The framework provides an outline for gathering information, which can help speed up the data gathering process.

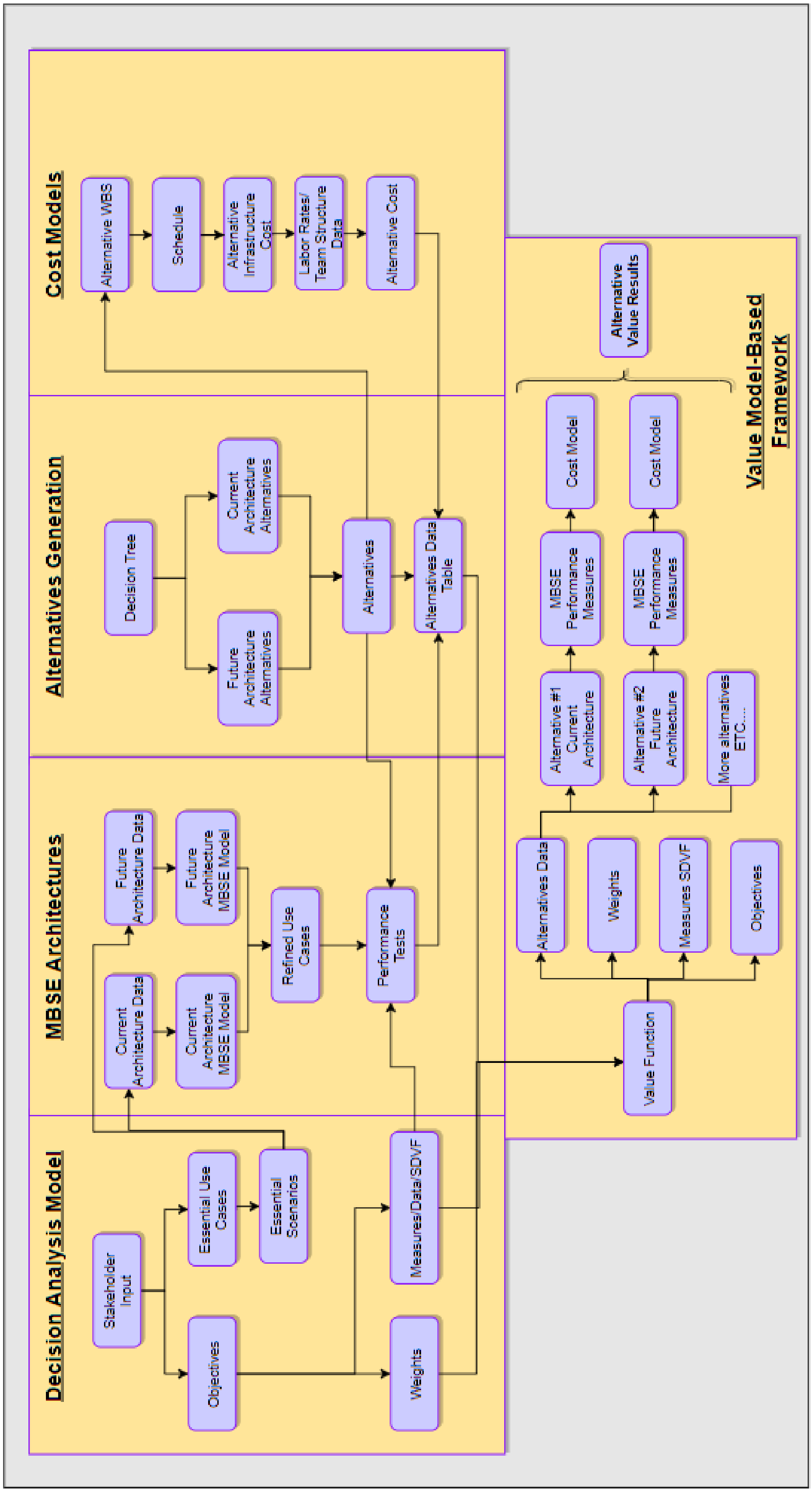

Decision Analysis Model
Stakeholder Input
Objectives
Essential Use Cases
Essential Scenarios
Weights
Measures/Data/SDVF
MBSE Architectures
Current Architecture Data
Future Architecture Data
Current Architecture MBSE Model
Future Architecture MBSE Model
Refined Use Cases
Performance Tests
Alternatives Generation
Decision Tree
Future Architecture Alternatives
Current Architecture Alternatives
Alternatives
Alternatives Data Table
Cost Models
Alternative WBS
Schedule
Alternative Infrastructure Cost
Labor Rates/ Team Structure Data
Alternative Cost
Value Function
Alternatives Data
Weights
Measures SDVF
Objectives
Alternative #1 Current Architecture
Alternative #2 Future Architecture
More alternatives ETC....
MBSE Performance Measures
MBSE Performance Measures
Cost Model
Cost Model
Alternative Value Results
Value Model-Based Framework

Figure 3.2: Linking the Component Models of the Value Model-Based Decision Framework

Figure 3.2 shows how the component models are linked and feed information into each other. It is important to note these dependencies and be prepared to update them as new information becomes available. In order to move along in the process initial assumptions can be made, but it is important to have those assumptions validated by the decision makers and stakeholders. The end result is to have a value associated with the architectures being compared. These values will help the decision maker know based on their priorities and requirements which is the best approach.

One piece of data that is very important to the framework are use cases. In order to evaluate the performance of the architecture it is important to test using one or more use cases. Throughout the process use cases and scenarios are used to test the alternative architectures. The initial use cases derived from the objectives and discussions with the decision maker are essential but might not include enough detail and could change later. The use cases derived from the current architecture represent the way things are currently done, which may include some of the unknown functions and details not yet captured. However current use cases will not capture future ways of operating, which is why future use cases are needed. Thinking outside the of the current business or operating model, what are the functions the organization wants the new system to do? The refined use cases take into account the Essential, Current, and Future use cases. They are the use cases used to evaluate the architectures on their performance.

3.1.1 Framework Setup

Framework Prerequisites Before embarking on the process of using the framework to help determine the best architecture design it is important to make sure the prerequisites are met. In order to use this process there should be enough time to gather all the information, build and analyze the models. It will take time to complete the process and the decision maker should allow the time to make sure it is done correctly. A decision maker should be identified and have the authority to make decisions on the project. The decision maker should be supportive of the process, be accessible, and provide help getting stakeholders, information, and resources. Many of the steps can be completed with a variety of tools but it should be determined up from if the tools required to do each step are available. The project should meet these criteria before going further into the process.

If the criteria are not met this framework will not be useful. Projects that require a really quick decision might only have time to use some parts of the framework. Projects where the decision maker doesn't understand or value the process should not use this framework. Projects where data or tools are not available will struggle to use this framework.

Setting Expectations with the Decision Maker Once the project has been determined as a good fit for this framework it is important to set expectations with the decision maker and go over the framework process. Each phase should be explained to the decision maker including the data requirements, tool requirements, the model outcome, and the decision maker's involvement. The decision maker should understand models are representations of reality, the better data and more time given to create and analyze the models the better the outcome will be, but there is no such thing as a perfect model. It should be stated to the decision maker that part of the benefit of this process is to help them understand their decision making and values. The process might not deliver the outcome they are expecting or wanting and they should be prepared to refine their preferences.

Understanding the Decision Maker It is important to understand the decision maker or decision makers and the organization they are making decisions for. In order to help them make decisions you first have to determine their style of decision making. Do they make decisions rationally with data? Do they make quick instinctual decisions or do they wait until more data is available? Understanding how they are used to making decisions will help determine the best way to provide them information, explain results, and how much time each part of the process might have before the decision maker wants to move to the next phase.

Determine what methods of communication work best for the decision maker. Do they prefer briefings, emails, phone calls, how often can they meet, and for how long? This will also help determine what information to present when meeting with the decision maker. Politics in an organization can influence a lot of decision making. Understanding conflicts with other groups in an organization can help make sense of unknown values and objectives. Decision makers may have hidden objectives they don't realize they have that influence their decision making. It can take several sessions with a decision maker to uncover unknown values. For the first couple of discussions focus not just on collecting data about the problem, but also trying to understand the decision maker on a deeper level.

Understanding the Decision Problem The first goal is to understand the decision problem the decision maker needs to make. It is not always what they tell you at first. Decision makers may communicate a problem in many ways and the challenge is to understand that problem and the context around it. They may be organizational influences that make the decision problem complex. There could be unique technical challenges to understand. It may take a couple conversations to make sure the problem is understood correctly and the decision problem is correctly framed.

Determining Resources The next several sub-sections, 3.1.2 to 3.1.8, will be outlining the models and the resources they require. Some models have several options for using tools to implement them. Before beginning the modeling process it is important to identify which tools will be used to build the models and to make sure they are available. Tools should also be chosen which are compatible with the organization. They might have a suit of tools they prefer or are familiar with. Choosing a tool that fits the organization will keep the decision maker focused on the results of the model instead of the tool that built the model.

Knowing how many people are available to help work the project is another resource that should be determined up front. The earlier roles, responsibilities, and tasks can be assigned and planned for. Adding people to the team later in the process can be a hindrance and not a benefit. Knowing what funding is available as a resource for the project can provide the team flexibility in how to approach the problem. Funding can be used for tools, pilots, data, or bringing on more people.

Setting the Schedule To acquire the data for the models several engagements with the decision maker and stakeholders will need to occur. The meetings should be focused on keeping the decision maker and stakeholders informed and obtaining useful information for the modeling effort. In person meetings are best, but emails or meetings with a representative will keep the project moving forward. These models can be adjusted as new information becomes available or decision makers change their minds. Outlining which engagements need to happen and what data is expected to be collected from each one will help keep the project and meetings focused. Below are a list of meetings that need to occur and what data should be collected and conveyed in each.

1. **Expectation Meeting:** Go over process, time constraints, resources, schedule, deliverables, and a communications plan. Set expectations with the decision maker of what the realistic outcomes will be. Get an understanding of the problem that needs to be solved.

2. **Objective Elicitation:** Discussion of organization's objectives and goals for the system. Confirm understanding of the problem being worked. Gather information to understand the context of problem, goals, and decision.

3. **Objective Hierarchy Feedback:** A meeting to verify the hierarchy is correct. Also discuss each of the initial ideas for measurements of the objectives. Go over the potential data sources, data range, and single-dimensional value functions.

4. **Weighting the Objectives:** Determine the different weights for each of the objectives. This could be from a single decision maker or a group discussion. Multiple methods could be used to gather different sets of weights that can be used later when analyzing the results.

5. **Identify Current Architecture Stakeholders:** Determine who are the users are of the system, who the owners are of each system, and other key stakeholders such as

security.

6. **Meet with the Stakeholders about the Current Architecture:** Obtain information about the current system's use cases, system infrastructure, data, communication between systems, access procedures, data security, licensed software, and cost. Note: This meeting could be broken into several meetings with different stakeholders or information could be provided by email or other forms of documentation.

7. **Meet with the Stakeholders about the Future Architecture:** Obtain information about their future use cases and desired functionality. Discuss issues with the current architecture that need to be addressed. Talk through any new constraints the stakeholders have identified for the new system. Note: This discussion may be combined with the previous meeting about the current architecture to save time.

8. **Feasibility and Alternatives Discussion:** Meeting with the engineering and/or security staff on the feasibility of the future architecture design. This meeting is an opportunity to get feedback from stakeholders on the proposed solution and identify any issues that need to be addressed. This is also an opportunity to gather information on a transition plan between the current architecture and future alternative ones.

9. **Decision Maker Update on Architectures:** Meeting to provide the decision maker information on architecture alternatives. It is also an opportunity to bring up any issues or concerns and receive feedback.

10. **Cost and Resource Discussion:** A discussion with the decision makers and stakeholders on the costs of the current architecture to make sure these cost are accurate. This should include costs of the additional resources it takes to complete a task currently using the architecture. The discussion should also cover estimations for the future architecture. The decision maker should be informed on build, transition, and operations costs. The goal is to receive feedback from the decision maker on whether these costs are within their range and if they have any additional resource information to provide.

11. **Framework Results Discussion:** This meeting is presenting the final results to the decision maker. It should summarize the work that's been done and all the assumptions and decision that have been made. The presentation should explain any areas of uncertainty. The decision maker should provide feedback on if these results make sense to them and if there are pieces of information missing.

3.1.2 Developing the Decision Analysis Model

Model Description This is the primary parametric model used to simulate the decision making. The goal is to create a value-function that measures architecture solutions on how well they meet the values of an organization.

Data Collection Data needed for this model are the stakeholder's end-objectives and means-objectives. The end-objectives are the top level goals for the system and the means-objectives are sub-goals on the way to achieving those end-objective goals. The objectives are organized in a hierarchy to show their level-relationships or decomposition. Weights for each of the objectives are needed for the model. The weights represent the importance of each objective in comparison with the others from the perspective of the decision maker. The final piece of data are the measures for each of the end-objectives. The measures are calculated using performance and cost data from the architectures and single-dimensional value functions which normalize the results. Each end-objective needs a measurement value that is preferentially independent from the other measurements and can be used to compare multiple alternatives. Most of these data points will come from the MBSE architecture models.

Tools At this stage in the modeling process a diagram tool is all that is needed. Having a tool for brainstorming and documenting objectives is useful. This could be done with a whiteboard or paper and documented later. If weights are calculated at this stage excel could be used.

Model Steps

1. Objectives are elicited from the organization's stakeholders. The objective hierarchy is formed and is communicated to all the key stakeholders to make sure everyone agrees it represents the organization.

2. Weights are determined for each of the objectives. This helps to establish the order of importance and how important each objective is compared with the others.

3. Methods for measuring the objectives are identified. Data is captured using other performance test models and cost models.

4. Single-Dimensional Value Functions (SDVFs) are selected to model the preference of the decision maker for each of the objectives. The data, data ranges, midpoints, and breakpoints are captured for each of the SDVFs to evaluate and normalize the data within the hierarchy.

5. Form the value function in a tool or compute the value function manually. Using the weights, the SDVFs, and the alternatives data.

6. Document the essential use cases that will be evaluated later against the current and future use cases.

Model Outcome The objective hierarchy is an artifact that visually represents the goals of the organization for the system. Essential use cases should be derived and documented as a result of the stakeholder objective elicitation. The model outcome won't be realized until the very end, which is described in the Combining Models Section 3.1.8.

3.1.3 Developing the Current Architecture Model

Model Description The current architecture represents the status quo. It describes what the current systems are, how they are used, how they communicate with each other, and how they interface with the external environment. The model is comprised of several diagrams that represent the behavior and structure of the current system. It is a hybrid model that makes it easy to capture and retain information about the systems architecture [64].

Data Collection The data that will populate this model will need to be captured from stakeholders in the organization and documentation. It won't always be in a form that is directly translatable into the model and will need to be interpreted. To document where information came from the MBSE tools should allow for notes to be used.

Tools SysML is the predominant language used to describe Model-Based Systems engineering architectures [63]. The SysML models can be built in several tools as long as they adhear to the SysML rules. Cameo is a popular commercial tool used to create MBSE models [65]. It will be the tool used to develop all the MBSE models including the current architecture because of its ease of use and availability.

Model Steps The steps provide an order to creating the model, but it is more like a checklist. These are views that will help document and understand the current architecture. Create different diagrams and elements as needed to document the current architecture.

1. Define the structure of the architecture with a Block Definition Diagram (BDD).
2. Define the stakeholders and use cases.
3. Define the subsystem functional architecture.
4. Define the subsystem physical architecture.
5. Define the internal structure and item flow.
6. Define the behavior of the system with activity, sequence, and/or state-machine diagrams.
7. Create an OV-1 describing the general use case and external interactions.

8. Define the systems value parameters used for performance assessment.

9. Create parametric models describing how the value parameters will be assessed. These will be the diagrams that describe the performance tests. Depending on the tools capabilities they can be used to run simulations.

Model Outcome The model will be a combination of diagram views and elements that capture information about the system. The diagrams and elements should have sufficient detail to asses the performance of the system.

3.1.4 Developing the Future Architecture Model

The approach to the creating the future architecture model is slightly different but the types of diagrams and elements are mostly the same. The current architecture model is about documenting the current architecture and how it is used. Creating the future architecture is where architects, developers, and systems engineers can be creative.

Teams can start with a reference architecture like Chen et al. suggests [41]. They can start from scratch and use IDEF0 diagrams to gain a macro level understanding of the project and iteratively get more granular with IDFE1 diagrams and so on. Or they can start by documenting the use cases and stakeholders. Then start create activity, sequence, state-machine diagrams to understand the functions the system will need to provide. Those functions can then help the team determine what subsystems or components of the system are needed. There are many ways a team can start generating what the future architecture could look like. The MBSE model will help make it easy to explore alternative ideas, document the alternative architecture, adjust as needed to new ideas, and provide version control.

Model Description The future architecture is the architecture design that should meet all the functional and non-functional requirements the organization has defined, if possible. It should describe what the systems will be, how it will be used, how the internal communications will work, and how it will interface with the external environment. The model is comprised of several diagrams and element that represent the behavior and structure of the future system. It is a hybrid model that makes it easy to capture and retain information about the systems architecture [64].

Data Collection Data will be elicited from the stakeholders for use cases and constraints. Research on big data architecture frameworks to reference. Research to see what the latest technologies are and if they can be utilized in the architecture. All the data and research can be referenced and documented in the model using notes in the diagrams views.

Tools The same MBSE tools should be used for the future architecture.

Model Steps The steps provide an order to creating the model, but it is more like a checklist. These are views that will help document and understand the future architecture. Create different diagrams and elements as needed to document the future architecture.

1. Define the structure of the architecture with a Block Definition Diagram (BDD).

2. Define the stakeholder use cases.

3. Define the subsystem functional architecture.

4. Define the subsystem physical architecture.

5. Define the internal structure and item flow.

6. Define the behavior of the system with activity, sequence, and/or state-machine diagrams.

7. Create an OV-1 describing the general use case and external interactions.

8. Define the systems value parameters used for performance assessment.

9. Create parametric models describing how the value parameters will be assessed. These will be the diagrams that describe the performance tests. Depending on the tools capabilities they can be used to run simulations.

Model Outcome The model will be a combination of diagrams and elements that capture information about the system. The diagrams and elements should have sufficient detail to asses the performance of the system and support the team who would be building the system.

3.1.5 Developing the Performance Tests

Model Description The performance tests are analytical models or simulations that measure the performance of the data architecture against the objectives. They are the linking models that turn the information collected in the MBSE models into data that can be used in the value function's SDVFs. They are built using the refined use cases, which make it so all architectures can be evaluated fairly. There should be a performance test for each of the objectives. The Decision Maker should agree that these tests meet there expectations for measuring performance.

Data Collection Essential, current, and future use cases are used to determine the refined use cases and test scenarios. Information form the MBSE architectures are used to understand what can be tested.

Tools Excel or scripts can be used to run the calculations. Depending on the objectives and use cases other tools might be available to test the performance of the architecture. For example there could be security tools that scan systems for vulnerabilities and the data from those tools could be used for the SDVFs.

Model Steps

1. For each objective identify a use case that can be used to evaluate it.

2. Determine what data will be needed from the MBSE models.

3. Determine the performance test equation or scoring method

4. Evaluate the architectures against each of the tests and collect the data ranges.

Model Outcome The results should be a table of data for each of the architectures on how they scored against each of the performance tests. The ranges for the possible outcomes will need to be captured as well. This data will feed into the SDVFs.

3.1.6 Generating Alternatives

Model Description Through the process the stakeholders and decision makers may float more ideas on different architecture approaches. This descriptive model is a way of capturing the "What if" alternatives that can be analyzed against the framework. The modeling process begins with only two alternatives in mind, the Status Quo Current Architecture and the Future Architecture, but as more information is collected and revealed other options will emerge. This model is a descriptive view of the options the decision maker is choosing from and the decisions they are making. The alternatives should always include the Status Quo option to make sure the other alternatives are going to deliver more value than the current state.

Data Collection Data on generating the alternatives comes from discussions with the Decision Maker. They may invite other stakeholder into the conversation. Later on more data can be captured from the other decision analysis, MBSE Models, and cost models.

Tools Because this is a conceptual descriptive mental model, any tool can be used that helps create pictures like trees and documents them. Microsoft PowerPoint or Visio, Draw.io, or any of the open source tools. This can initially be done on a white board or paper and then translated to a tool later.

Model Steps This picture provides the decision maker a view of where they are currently at in the process. The alternatives can be discussed with the decision maker once enough information has been gathered on their objectives and an understanding of the current and future architecture are mature enough. This visual can help decision makers get a deeper understanding of what their choices are and what decision they are trying to make. The diagram can be added to as information becomes available. The steps below are iterative.

1. Brainstorm alternative options with the decision maker and stakeholders.
2. Diagram the decision making with a decision tree and present to the decision maker for feedback.
3. Gather data on the alternatives by creating new MBSE models or modifying the existing ones.
4. Evaluate alternatives data against the performance tests.
5. Calculate the single dimensional value functions.
6. Build cost models for the alternatives.
7. Calculate the value function.
8. Collect all the alternatives data and add it to the diagram. The alternative's values, cost, schedule etc.

Model Outcome The initial results are a list of alternatives to collect information on and a visual representing the decisions. This model is what drives the data collection and analysis. It provides the decision maker an overview of where they are at in the process and what the next steps are. Once the analysis is complete the picture can be updated to include all the information the decision maker needs in order to make a decision.

3.1.7 Developing the Cost Models

Model Description The cost models should represent the total investment of each architecture. It should include development costs, maintenance costs, infrastructure and/or licensing costs. Each architecture will have a WBS, schedule, labor rates, labor rates for the teams, and infrastructure cost. The total cost will depend on the length of time it takes to develop the alternatives. The model should also account for O&M costs to show decision makers what their costs could be several years in the future.

Data Collection Data for the infrastructure costs include physical architecture costs like servers, data centers, rented rooms, electricity, cloud costs, etc. Labor information should be collected or estimated from existing contracts, employee rates, or average hourly rates based on job titles [66]. Information needed to create a work breakdown structure for each alternative is needed. Estimations for how long each of the WBS tasks is needed. If using commercial products licensing fees for software should be captured.

Tools There are many industry tools that can help create schedules, work breakdown structures, and/or estimate infrastructure costs. Some like Microsoft Project can do all the calculating and scheduling. A combination of Excel and Draw.io are the tools used for this analysis because of it's flexibility and availability.

Model Steps

1. Create the Work Breakdown structure for each of the alternatives.
2. Develop the schedule.
3. Create the teams with labor rates.
4. Calculate the Labor Costs for each alternative.
5. Calculate the monthly infrastructure and licensing costs for each alternative.
6. Calculate the total cost for each alternative, based on the monthly rates and schedule.

Model Outcome The model output should be a cost estimate that includes all aspects of the architecture. It should be a model that can easily be adjusted as more information is provided.

3.1.8 Combining the Models

Model Description The final parametric value model takes data in from the other models and provides a resulting value. The framework for the final model is the decision analysis value function. The data that feeds into the value function are the alternative's performance and cost data, which come from the other models previously discussed. The data is then normalized by the Single-Dimensional Value Functions and each alternative receives an SDVF score for each of the measures. The weights are another data input that makes the model specific to the organizations preferences. The additive value function calculates the final scores using the SDVF score and weights. The results can then be ranked based on the final value scores.

Data Collection The data comes from the alternative data table which contains the performance and cost information for all the alternatives. This data is what will be normalized by the SDVFs. Weights are information that can be collected again if necessary.

Tools There are decision analysis tools that can be used to help calculate the value function and keep track of the measurement data, weights, and measurement functions. It can also be done manually by hand or in excel. Logical Decisions is a tool commonly used to develop this model because of its organizational benefits, calculations, and sensitivity graphs. "As of January 1, 2021, Logical Decisions is now freeware. Version 8.0 is now free of charge and contains all of the features of the professional, group and portfolio versions and does not require a license key. It is released under the MIT license [67]."

Model Steps

1. Create the alternatives data table with data from each of the models.
2. Evaluate the alternatives against the SDVFs.
3. Calculate the weights
4. Evaluate the alternatives using the additive value function, the SDVF scores and the objective weights.

Model Outcome The value function is the executable model, which should output a value that represents how an architecture measures against the objective hierarchy. It should be comprised of the the weights of the objectives and the single-dimensional value functions for measuring each of the objectives. The model will provide a rank order of the alternatives based on their final calculated value.

3.2 Experiment Deign

Experiment Overview The experiment is meant to be an example and demonstrate how the Value Model-Based Decision Framework can be applied to an organization wanting to update their data architecture. To conduct the experiment a hypothetical organization is defined. The organization will have common characteristics of many large organizations including their data problems. The organization will have a current data architecture, strategic goals for the new architecture, stakeholders, a decision maker, and a project team to work their data modernization initiative. To help demonstrate the value of an Enterprise Data Architecture, the organization has selected a data science project to test both the current and future architectures. This will highlight how the current architecture struggles to support data science and the future architecture will improve this.

Terms There are some terms that need to be defined in the context of this experiment. Clarifying their meaning will help with the understanding for the example set up.

1. **Case Study** The organization for this example will be the presumptive case study. The organization is hypothetical, but has common structure, issues, and strategic goals as many organizations looking to develop an enterprise data architecture do.

2. **EDA Development Initiative** This group represents the effort the organization has approved and funded to plan and develop the Enterprise Data Architecture. The team will use the Value Model-Based Decision Framework to analyze the organization's current situation and help them make a decision on a path forward.

3. **Executive Dashboard Project** This project is a data science effort that will be used to demonstrate how well the current and future architectures support data sciences projects. It is a project that tests all aspects of the architecture and is a common project organizations undertake. It is the type of project that would struggle under the current architecture. The future architecture should have no problem enabling this kind of effort.

4. **Use Case** A use case will be a general task the architecture should enable. Both the current and future architecture will have use cases they support. Use cases don't have to just be supporting functional users. They can also be supporting other external systems.

5. **Scenario** A scenario is a specific activity performed of a use case. It will detail the user, the activities, the data, and the outcome. It would be a specific example of how a use case would work.

3.2.1 Defining the Hypothetical Organization

Organization Overview A large organization has been around several years and has accumulated several IT systems to support different purposes in the organization. It has over 1000 employees, an annual budget of over 1 billion, and a full-time IT Department [68]. As their organization has grown new systems have come online to support different areas of the organization. Each IT data system has it's own team maintaining it with a program manager (PM) being the owner of the system. The PM manages the engineering team and is the decision maker for the data system. The IT department has it's own security team that is responsible for making sure all IT systems are secure and designed to the organization's security standards. The security team has security engineers who review systems and security analysts who monitor existing system. The organization has other departments who provide requirements to the IT department for the systems that support their needs.

Leadership in the organization recently became interested in some of the new advances in technology using data science. Across the organization they are looking for opportunities to improve their mission with data science methods. They hired experienced data scientists to work problems, but they ran into issues with getting access to the data they need for the analysis. The organization's data are in behind disconnected data systems, which are hard to discover and get access to. Departments with data science projects across the organization start levying requirements on the IT Department to make the data more available. The organization's leadership recognizes this requires a new architecture to support the future data demands. They have stated their new strategic goal is to bring the data systems together, which will improve their ability to derive information and perform advanced analytics.

Current Challenges The current architecture is segmented into several single-purpose built systems which causes multiple issues. All the systems were designed separately and are managed by separate teams in the organization. They are not designed to support data science efforts across the organization. They don't have consistent data access procedures, there is no data catalog for data scientists to discover data, and the systems weren't built for the processing demands data scientists require. Data science teams spend a lot of time waiting to get access to data systems or waiting for data to be given to them. Data science teams don't know where to host new data sets they create. Multiple IT systems come with additional overhead costs such as separate teams to sustain them and additional burden on the security team to review and monitor multiple systems. Many of the data systems haven't had their security upgraded and may have vulnerabilities that need to be addressed. The IT Department doesn't have a good method for determining what data systems they can get rid of in order to reduce costs, so the IT budget keeps growing. Across the organization time and money is being wasted because the current architecture has grown to an unmanageable size and is not aligned with their future goals.

Strategic Goals The organization is now looking to modernize their architecture to support their new data science initiatives. They recognize the strategic importance of updating their infrastructure to support the current and future data requirements of the organization. They recognize this is also an opportunity to reduce costs, reduce security risks, and improve resilience of their data systems. They are not looking to make a quick fix but to reevaluate how their organization operates with respect to data and make the whole organization more efficient. The organization is committed to changing their culture around data and willing to make the investment in a new architecture.

EDA Development Initiative The organization has created a new team to develop a solution to their data challenges. The Enterprise Data Architecture Initiative is a project under the IT Department that will evaluate the current data architecture and develop a design for the future data architecture. The organization has budgeted for them to work a year to complete their analysis and present a recommendation on the path forward. The team will have all the tools they need to conduct the analysis. The team has the full support of the organizations leadership and will be able to meet with all team across the organizations to gather data, use cases, and requirements.

Executive Dashboard Project One of the high profile data science efforts in the organization is the Executive Dashboard Data Science Project. The team has been thus far unable to build a dashboard that meets the requirements of the executive because they are unable to get access to the data they need, they have not been able to connect the dashboards directly to the data systems for real-time information, and they don't have a place to store new or merged data causing them to have to rerun processing scripts everyday. Leadership has identified this data science effort as the initial stakeholder the Enterprise Data Architecture should support.

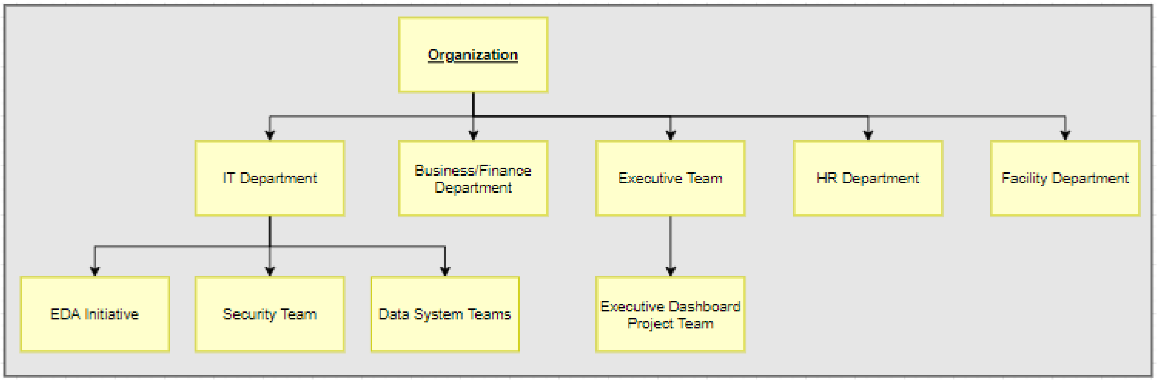


Figure 3.3: Example Organization Hierarchy

Organization and Team Structure The organization is a hierarchical structure with departments and teams. The graphic 3.3 shows the different departments and teams that will play a role in the experiment. There are more teams and departments that would need to exist for the organization to operate but are not involved with the EDA initiative and don't need to be defined.

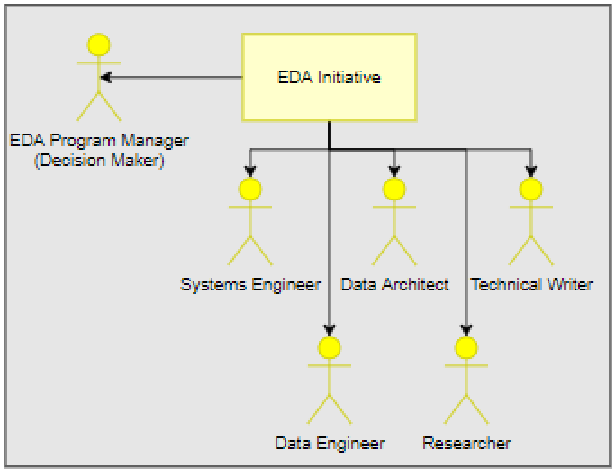


Figure 3.4: Example Enterprise Data Architecture Initiative Team

Figure 3.4 is a picture showing who is on the EDA initiative team. The enterprise data architecture (EDA) team will have a primary decision maker for the EDA effort, whom will be the project manager. They are responsible for staffing the team, providing resources and direction. The systems engineer will be responsible for the framework model and analysis. They are the ones who will build the value based decision-model to help the EDA Project Manger determine the best path forward. The systems engineer will rely on other team members to help collect data, document information, and get expert opinions on technologies or specialty areas.

Figure 3.5 is a picture showing who is on the executive dashboard data science project. The executive dashboard team will be the primary functional stakeholders the EDA team will engage with. Their team is comprised of the major data science skill-sets. The data

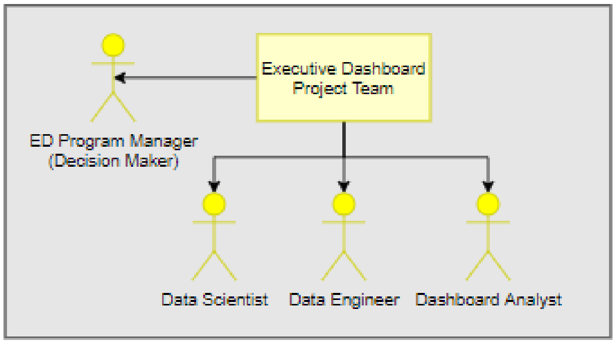

Executive Dashboard Project Team
ED Program Manager (Decision Maker)
Data Scientist
Data Engineer
Dashboard Analyst

Figure 3.5: Example Executive Dashboard Team

scientist will need access to large amounts of data, large amounts of compute, and advanced modeling tools. The data engineer will be more interested in developing new databases and schemas to support the effort. The dashboard Analyst will be more focused on the visual presentation of data.

3.2.2 Executive Dashboard Project

Executive Dashboard Description An executive for this large organization would like to have a dashboard that displays real-time information about the organization at the executive level. The executive wants the information in the dashboard to always be up to date and available on all their devices. The executive doesn't want raw information, they want information displayed that can provide quick insight into their questions, meaning it should be a combination of data sources that have been modified and analyzed. An example of the top three questions executives want the dashboard to answer are:

1. How is the budget being spent?

2. Where are employees and contractors located and what projects are they working on?

3. Are the top projects of interest in the organization on track to deliver and resourced properly?

In order to answer these questions multiple data sources need to be joined, filtered, and analyzed.

Project Overview In order to full-fill the Executive Dashboard requirements the Executive Dashboard team will need access to human resource (HR) data, facility data, project data, and finance data. The team needs an environment to extract data to where it can be manipulated and analyzed using their advanced analytics tools. The team will need to create a new data sets for the dashboard to pull from in order to reduce the amount of data manipulation in the dashboard. The scripts used to modify the data should be turned into data pipelines which populate the new data source from old data sources with the custom modifications. The data science team will need a long-term solution for storing the new data and data pipelines. The data science team will also need to have the dashboard connect to the data sources so the dashboard can be updated in real-time.

EDA Initiative and Executive Dashboard Team Collaboration To ensure the enterprise data architecture can meet the needs of the executive dashboard's team they will need to work closely together. The EDA team will rely on the executive dashboard team to provide use cases which will then be translated into specific scenarios that can be used to evaluate the current and future architectures. The EDA team will use the executive dashboard team as functional testers and have them validate functional requirements.

3.3 Experiment Models

Models Overview Each of the models will be built in the phased approach as illustrated by figure 3.2. The Decision Analysis model is formed first by identifying the objectives, weights, essential use cases, and the Single-Dimensional Value Function for each of the measures.

In the next phase the MBSE models for the current and future architectures. Data will need to be collected and the models built. Each model should include a parametric model that will be used for evaluating performance of the architecture. Performance tests will then be developed to evaluate both of the architectures against the performance measures.

In the next phase the alternatives will be generated after presenting the MBSE architectures to the decision maker. While figuring out how to transition from the current architecture to the future one, different alternative approaches will be discussed. The decision maker state which alternative's they are interested in evaluating. The alternatives will be evaluated against the performance tests and the performance metrics result will be added to the alternative's data table.

The cost models will be built for each alternative. Data will be collected from each of the architectures to help with the pricing for the labor and materials. A schedule is will be developed based on what parts of the architecture are built or modified. The final result will be an estimation price range for each alternative architectures.

The final phase combines all the models to get our final value results for each alternative. A data table will capture each of the measure values for the alternatives that will be input into the decision analysis model. The value function will evaluate each alternative using the single-dimensional value functions and the weights. The results of the model will be discussed in the Section 3.4.

3.3.1 Framework Setup

Framework Prerequisites This project meets all the prerequisite requirements. The organization is fully on-board with providing the resources and time to conduct the research and analysis needed to determine the best path forward for their future enterprise data architecture. A decision maker has been identified for the project and has been given the authority to make the decisions on behalf of the organization. The decision maker is supportive of the process, is a part of the team, and is well connected in the organization so they will be able to provide the stakeholder connections and tools.

Setting Expectations Expectations have been set with the decision maker. They are aware of the steps in the process, data, and tools that will be needed, and their role in the process. They have understanding of the time it will take to run through each phase. They have and understanding of the outcomes to expect and the limitations of the process.

Decision Maker The decision maker is a rational decision maker who is interested in what the data and models will provide. They are available often for feedback via email, quick meetings, or more formal engagements. The decision maker is well connected with the organization and is aligned with the strategic objectives of the organization. They are good communicators and do not have any hidden objectives.

The decision maker is sensitive the the goals and desires of their colleagues throughout the organization. They listen to security's concerns, the Executive Dashboard team's needs, and the data system owners experiences. Because the decision maker represents the organization they are influenced by other components of the organization. This is challenging when there are competing values, but the decision maker does their best to decide what is more important for the organization as a whole.

Decision Problem The problem the organization has is their current architecture does not support data science efforts, it is expensive, and there is a lot of risk. The decision problem is to decide on what enterprise data architecture design will address those issues for the best price. Alternatives will be generated and the decision maker will want to determine which is the best alternative for the organization.

Later on in the process once the current and future architectures have been presented, the decision problem revolves around whether to build and enterprise data architecture or just improve the current architecture. Subsection 3.3.5 will go over the decision problem in more detail.

Resources The organization has fully resources this project with the people and tools needed. The people with the appropriate skillet have been assigned to the team and they understand their roles and responsibilities. The team has identified using Cameo Architecture, Draw.io, and Excel to build and create the models. These tools align with the organization and the decision maker is familiar with them.

Schedule The following are a list of essential meetings that were conducted with the decision maker or stakeholders through the process in order to obtain information or validate the work.

1. **Expectation Meeting:** This meeting was conducted with the EDA project manager who is the decision maker for this project. The Value Model-Based Decision Framework was explained including the process, the requirements, the decision maker's required involvement, and the expected outcomes.

2. **Objective Elicitation:** This meeting was conducted with the EDA project manager. The team talked about the organizations values, the desires for the future system, and constraints needed on the system like system security.

3. **Executive Dashboard Team Interview:** Meeting to understand the specific needs of the Executive Dashboard team to capture use cases and specific scenarios.

4. **Objective Hierarchy Feedback:** At this meeting the draft objective hierarchy was presented to the decision maker for feedback.

5. **Weighting the Objectives:** At this meeting the EDA project team walked the decision maker through the swing weight process to collected the weights of the objectives.

6. **Identify Current Architecture Stakeholders:** This meeting was to identify key stakeholders in the organization whom have information needed for the models. Systems containing data the Executive Dashboard Team is interested in were listed as well as security stakeholders. The EDA decision maker was able to provide all points of contact.

7. **Meet with the Stakeholders about the Current Architecture:** Several meetings were conducted with the data system teams to understand their current data security, system structure, process for getting access, cost of the system, number of hours worked on the system, and other data needed for the models.

8. **Meet with the Stakeholders about the Future Architecture:** Several meetings were conducted with the decision maker, data systems teams, security stakeholders, and the Executive Dashboard team to understand what their desires for a future architecture were.

9. **EDA Feasibility Discussion:** This meeting was to present alternative future designs to the Decision Maker for feedback. The outcome of this meeting was to gain approval and feedback on the initial EDA design.

10. **Decision Maker Generate Alternatives:** The meeting was to provide the decision maker an update on the current and future architectures and how they will be measured. The performance tests were discussed along with what alternatives the decision maker was interested in. The goal was to validate the performance tests, get a list of alternatives, and determine the SDVF preferences from the decision maker.

11. **Cost and Resource Discussion:** This meeting with the decision maker was to discuss the results of the cost models including the work breakdown structures, bills of materials, labor rates, and roadmaps. The goal was to get feedback on the cost model results and artifacts.

12. **Framework Results Discussion:** This meeting was to discuss the final results of

the value function for each of the alternatives. Different views were presented to the decision maker describing how each models performance compared. The decision maker provided feedback for some what-if scenario analysis.

13. **Framework What-if Results Discussion:** The results for the previous meetings feedback were presented to the decision maker. The new results were compared with original results.

3.3.2 Decision Analysis Model

Data Collection The data for the decision analysis model comes from the decision maker, stakeholders, the MBSE models, and cost models. The objectives were the first piece of data collected from the decision maker during the Objective Elicitation Meeting. Data for the essential use case and scenarios were gathered from the Executive Dashboard Team and Decision Maker. Weights were collected from the decision maker during the Weighting Objectives Meeting. Data and initial data ranges for the measures comes from the MBSE Architecture models in the current architecture subsection 3.3.3 and the future architecture subsection 3.3.4. Final data results for the alternatives will come from the performance tests subsection 3.3.6 and system cost models in subsection 3.3.7.

At the end of the first phase, the actual data results were not needed, only an understanding of what the measures are, some ideas on how the data can be collected, and what the decision maker's preference direction is for each of the measures. The actual data results and SDVFs will be brought together as the full model in subsection 3.3.8.

Objective Hierarchy Figure 3.6 is the final Objective Hierarchy, which was approved by the decision maker. The fundamental objective is to decide on an Enterprise Data Architecture. The top three end-objectives are Minimize Risk, Maximize Enabling Data Users and Minimize Cost.

Minimize Risk was decomposed into two means-objectives, Maximize System Resilience and Maximize Data Security. Minimize Risk is a priority to the decision maker because it

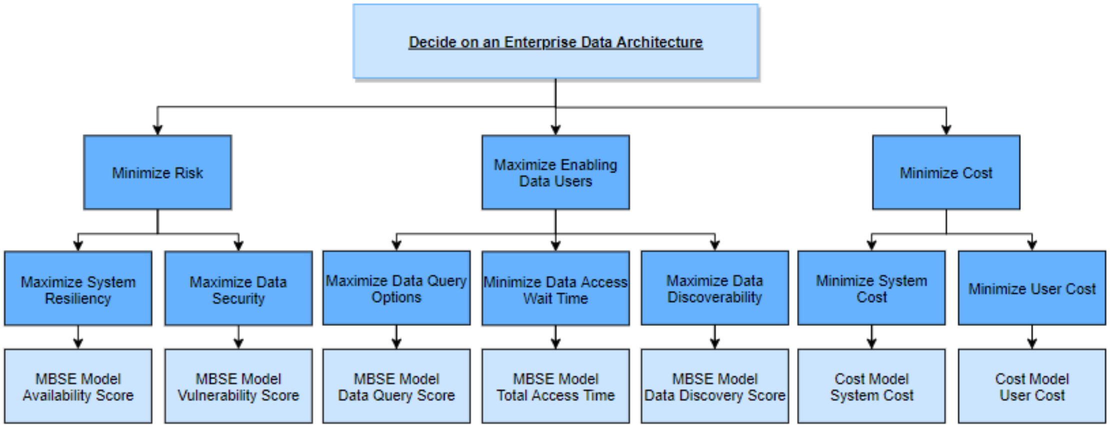

Decide on an Enterprise Data Architecture
Minimize Risk
Maximize Enabling Data Users
Minimize Cost
Maximize System Resiliency
Maximize Data Security
Maximize Data Query Options
Minimize Data Access Wait Time
Maximize Data Discoverability
Minimize System Cost
Minimize User Cost
MBSE Model Availability Score
MBSE Model Vulnerability Score
MBSE Model Data Query Score
MBSE Model Total Access Time
MBSE Model Data Discovery Score
Cost Model System Cost
Cost Model User Cost

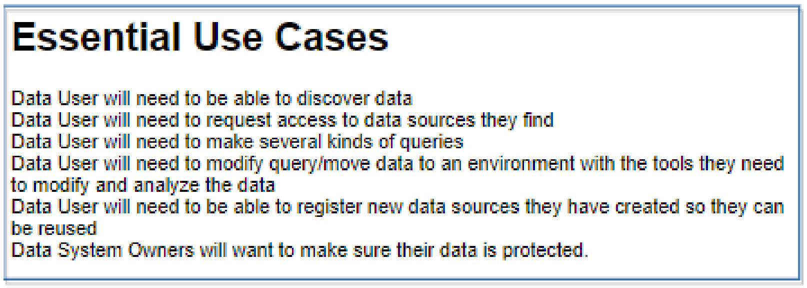
Figure 3.6: Example Objective Hierarchy

represents the possibility of negative outcomes that could occur and effect the organization negatively and they want to reduce the likelihood of that happening. Maximize Resiliency is a priority to the decision maker because they want to make sure that if a negative outcome were to occur the systems would withstand the negative impact and not fail. Maximize Data Security is a priority to the decision maker because safeguarding data from data spills, insider threats, and external threats is important to the stakeholders of the organization.

Maximize Enabling Data Users was decomposed into three means-objectives, Maximize Data Query Options, Minimize Data Access Wait Time, and Maximize Data Discoverability. Maximize Enabling Data Users is a priority because it represents the main function of the system. Maximize Data Query Options is important to the users of the system because it offers them more options for extracting the data they need. Minimize Data Access Wait Time is important to the users of the system because the more quickly they can get access to the data the need the sooner they can begin their other work. Maximize Data Discoverability is important to the user of the system because if they can't find data they can't request it or use it.

Minimize Cost was decomposed into two means-objectives, Minimize System Cost and Minimize User Cost. Minimize Cost is a looking at reducing the total cost of ownership for the system. Minimize System Cost is important because the organization wants to either reduce or maintain their current IT budget while supporting more requirements and getting more value out of their data systems. User Cost is important because it is a measures the reuseability of the system which should reduce duplicate work. The organization wants to make their data science teams more efficient and effective with long term solutions. The faster a data science team can complete their project, the cheaper the project will cost and more data science projects can be accomplished.

Essential Use Cases The essential use cases are an initial set of uses cases derived from discussions with the decision maker and the executive dashboard stakeholders. They provide an initial understanding of what functions are needed in the architecture. These functions are general and not specific to any project that is using the system, they represent what all projects using the system would need and provide context into developing the objectives. Figure 3.7 is the initial list of essential use cases.

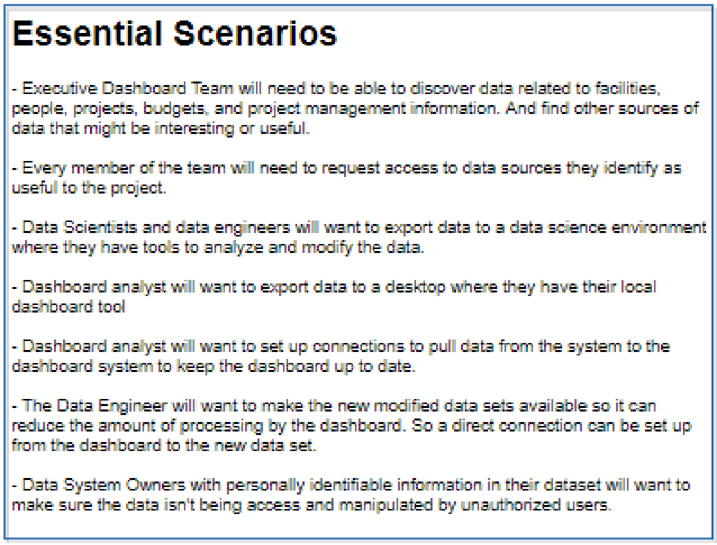


Essential Use Cases

Data User will need to be able to discover data
Data User will need to request access to data sources they find
Data User will need to make several kinds of queries
Data User will need to modify query/move data to an environment with the tools they need to modify and analyze the data
Data User will need to be able to register new data sources they have created so they can be reused
Data System Owners will want to make sure their data is protected.

Figure 3.7: Essential Use Case

Essential Scenarios The Essential Scenarios are specific workflows from the executive dashboard project. They specify what workflows are needed in order for them to complete their project. They specify what data is needed, how they want to query data, where they need the data to go so they can merge, analyze, and modify the data. These essential scenarios will be used to test the current and future architecture designs to determine how well each architecture supports these specific scenarios. Those performance measures will be the data sources for the measures in the value-hierarchy. Figure 3.8 are is an initial list of essential scenarios.



Essential Scenarios

- Executive Dashboard Team will need to be able to discover data related to facilities, people, projects, budgets, and project management information. And find other sources of data that might be interesting or useful.

- Every member of the team will need to request access to data sources they identify as useful to the project.

- Data Scientists and data engineers will want to export data to a data science environment where they have tools to analyze and modify the data.

- Dashboard analyst will want to export data to a desktop where they have their local dashboard tool

- Dashboard analyst will want to set up connections to pull data from the system to the dashboard system to keep the dashboard up to date.

- The Data Engineer will want to make the new modified data sets available so it can reduce the amount of processing by the dashboard. So a direct connection can be set up from the dashboard to the new data set.

- Data System Owners with personally identifiable information in their dataset will want to make sure the data isn't being access and manipulated by unauthorized users.

Figure 3.8: Essential Scenarios

Measures Each objective needs a measurement value. The measurement value will be the results of a Single-Dimensional Value Function (SDVF) that normalizes and compares how alternatives perform against an objective. The SDVFs are determined based on the data and preferences of the decision maker. Each of the following paragraphs will describe how the SDVFs were developed for each of the objectives.

Maximize System Resiliency To measure the resiliency of the system the decision maker indicated that making sure system's performance didn't degrade as data usage increased. The EDA team determined testing the system's availability performance over a year period would be a good measure to show how resilient the current and future systems are.

To come up with an availability value each architecture will need an estimate of the

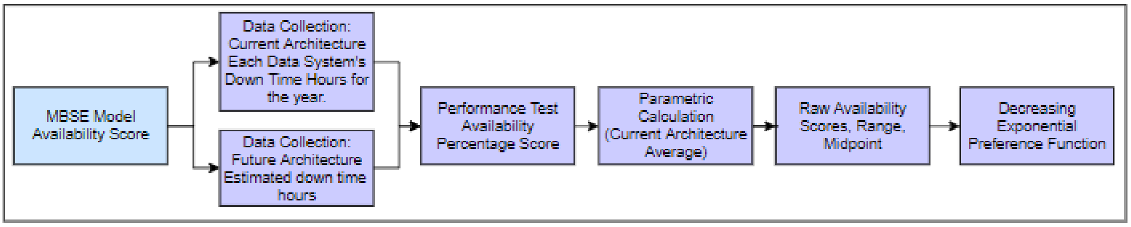

MBSE Model Availability Score
Data Collection: Current Architecture Each Data System's Down Time Hours for the year.
Data Collection: Future Architecture Estimated down time hours
Performance Test Availability Percentage Score
Parametric Calculation (Current Architecture Average)
Raw Availability Scores, Range, Midpoint
Decreasing Exponential Preference Function

Figure 3.9: Calculation Overview for Availability Score

amount of downtime that occurred over the past year. The information will be used to calculate the Average Availability Percentage Score for each architecture. Based on the possible availability scores a range will be determined. The decision maker determined that their preference changed over the range of values, so the decreasing exponential function was selected and a midpoint was chosen.

- **Data:** Performance Test Average Availability Percentage Scores
- **Function Type:** Decreasing Exponential Preference Function
- **Direction of Preference:** Maximize
- **Range:** XBest = 100, XWorst = 80
- **Midpoint:** 97%
- **SDVF** $V(X_i)$**:**

$$V(X_i) = \frac{1 - e^{\frac{(80 - X_i)}{-.044}}}{1 - e^{\frac{(80-100)}{-.044}}} \tag{3.1}$$

Maximize Data Security To measure the data security of the system the decision maker indicated that the architecture should have minimal vulnerabilities and follow secure system principles. To measure this the architectures will be evaluated for their vulnerabilities. In order to Maximize Data Security the SDVF will be minimizing the number of vulnerabilities.

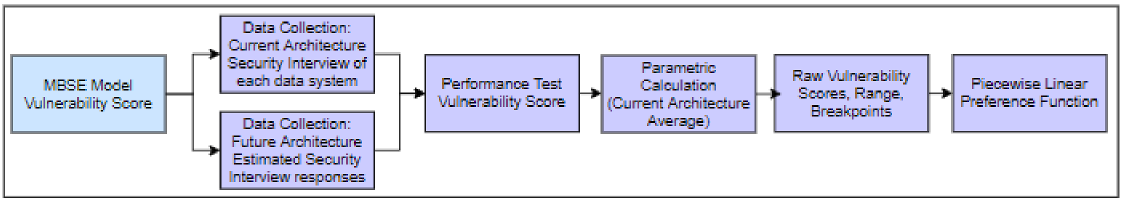

MBSE Model Vulnerability Score
Data Collection: Current Architecture Security Interview of each data system
Data Collection: Future Architecture Estimated Security Interview responses
Performance Test Vulnerability Score
Parametric Calculation (Current Architecture Average)
Raw Vulnerability Scores, Range, Breakpoints
Piecewise Linear Preference Function

Figure 3.10: Calculation Overview for Vulnerability Score

To come up with a vulnerability value each architecture will be assessed based on their responses to security questions. The information will be used to calculate the Average Vulnerability Score for each architecture. Based on the possible vulnerability scores a range will be determined. The decision maker's preferences for security were grouped into four ranges. They provided those ranges as breakpoints and values at each of those breakpoints so a piecewise linear function could be created.

- **Data:** Performance Test Average Vulnerability Scores
- **Function Type:** Piecewise Linear Preference Function
- **Direction of Preference:** Minimize
- **Range:** XBest = 0, XWorst = 40
- **Breakpoints:** V(10) = .9, V(20) = .6, V(30) = .2
- **SDVF** $V(X_i)$**:**

$$V(X_i) = \begin{cases} 1 + \frac{(.9-1)}{(10-0)} * (X_i - 0), & 0 \le x \le 10 \\ .9 + \frac{(6.-.9)}{(20-10)} * (X_i - 10), & 10 \le x \le 20 \\ .6 + \frac{(.2-.6)}{(30-20)} * (X_i - 20), & 20 \le x \le 30 \\ .2 + \frac{(0-.2)}{(40-30)} * (X_i - 30), & 30 \le x \le 40 \end{cases}$$

Maximize Data Query Options The decision maker indicated that the architectures should provide multiple options for users and other systems like the dashboard system and data science environment to query data. To measure this the architecture will be scored based on the type of query options they provide.

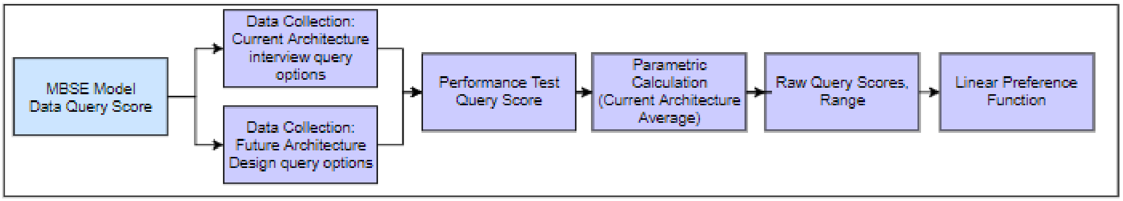

MBSE Model
Data Query Score
Data Collection: Current Architecture interview query options
Data Collection: Future Architecture Design query options
Performance Test Query Score
Parametric Calculation (Current Architecture Average)
Raw Query Scores, Range
Linear Preference Function

Figure 3.11: Calculation Overview for Data Query Score

To come up with an query score each architecture will be assessed based on their responses to questions about how data users can query and extract their data. The information will be used to calculate the Average Query Score for each architecture. Based on the possible query scores a range will be determined. The decision maker's preferences for query options was consistent over the range of values, so a linear preference function was chosen as the SDVF.

- **Data:** Performance Test Average Data Query Scores
- **Function Type:** Linear Preference Function
- **Direction of Preference:** Maximize
- **Range:** XBest = 30, XWorst = 6
- **SDVF** $V(X_i)$**:**

$$V(X_i) = \frac{X_i - 6}{24} \tag{3.2}$$

Minimize Data Access Wait Time To measure the data access wait time the decision maker indicated that the request process beginning with the initial request and ending with a notification that they now have access to the data is what matters to the users of the system. Each architecture's request process time will be estimated and averaged to provide a final score.

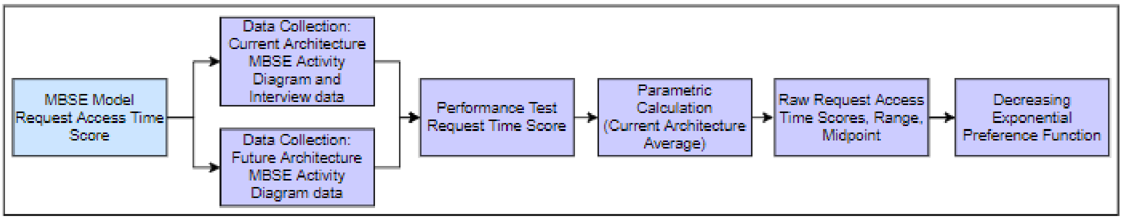

MBSE Model Request Access Time Score
Data Collection: Current Architecture MBSE Activity Diagram and Interview data
Data Collection: Future Architecture MBSE Activity Diagram data
Performance Test Request Time Score
Parametric Calculation (Current Architecture Average)
Raw Request Access Time Scores, Range, Midpoint
Decreasing Exponential Preference Function

Figure 3.12: Calculation Overview for Total Access Time Score

To measure this the architectures request process will be documented and each step of that process will be evaluated for how long it takes to complete. The total wait time will sum up each step estimated time. The information will be used to calculate the Average Request Time Score for each architecture. Based on the possible request time scores a range will be determined. The decision maker's preferences for the request time changed over the range. A decreasing exponential preference function was chosen as the SDFV and the midpoint was provided by the decision maker.

- **Data:** Performance Test Request Access Time Scores
- **Function Type:** Decreasing Exponential Preference Function
- **Direction of Preference:** Minimize
- **Range:** XBest = 1, XWorst = 14
- **Midpoint:** 3 Days
- **SDVF** $V(X_i)$**:**

$$V(X_i) = \frac{1 - e^{\frac{(14 - X_i)}{-3.068}}}{1 - e^{\frac{(14-1)}{-3.068}}} \tag{3.3}$$

Maximize Data Discoverability To measure the Data Discoverability the decision maker indicated that each data source in the architecture should have some method for finding out about it. Whether it's by word of mouth, or it has a listed API, or contact information. To measure this each architecture will estimate how long it takes each data source to be discovered using their different methods of discovery.

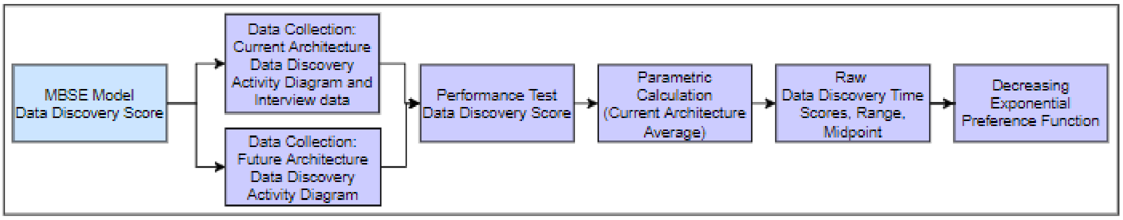


Figure 3.13: Calculation Overview for Data Discoverability Score

Each architecture has two potential methods for discovering data sources. Each architecture will provide information on whether the data sources can be discovered via search and an estimate of how many people in the organization know about the data source in order to determine how likely a data user would be to run into someone with the information they need. The time it takes to discover the system will be calculated by assigning a search time and estimating how much time it would take to find a data source based on word of mouth. The word of mount time will be estimated using a geometric distribution. The information will be used to calculate the Average Data Discovery Score for each architecture. Based on the possible data discovery time scores a range will be determined. The decision maker's preferences for the request time changed over the range. A decreasing exponential preference function was chosen as the SDFV and the midpoint was provided by the decision maker.

- **Data:** Performance Test Data Discovery Time Scores
- **Function Type:** Decreasing Exponential Preference Function
- **Direction of Preference:** Minimize

- **Range:** XBest = 1, XWorst = 16
- **Midpoint**: 2 Days
- **SDVF** $V(X_i)$**:**

$$V(X_i) = \frac{1 - e^{\frac{(16-X_i)}{-1.515}}}{1 - e^{\frac{(16-1)}{-1.515}}} \tag{3.4}$$

Minimize User Cost For the user cost the decision maker wants to know how expensive it is to execute a data science project with each architecture. The user cost will be assessed using the executive dashboard project as an example data science project.

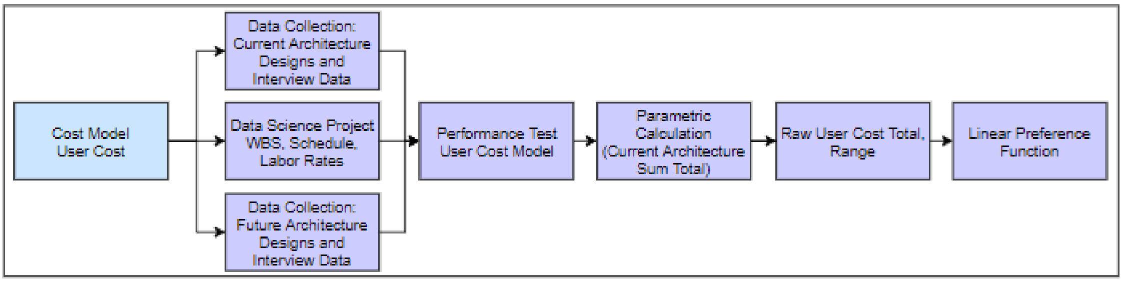

Cost Model
User Cost
Data Collection: Current Architecture Designs and Interview Data
Data Science Project WBS, Schedule, Labor Rates
Data Collection: Future Architecture Designs and Interview Data
Performance Test User Cost Model
Parametric Calculation (Current Architecture Sum Total)
Raw User Cost Total, Range
Linear Preference Function

Figure 3.14: Calculation Overview for User Cost

The executive dashboard data science project will be used to calculate the cost of the project under each architecture. The cost will take into account the length of time it takes to complete the project using each architecture and the labor rates for the team. The information will be used to calculate the User Cost Score for each architecture. Based on the possible user cost scores a range will be determined. The decision maker's preferences for the user cost scores was consistent over the range of values, so a linear preference function as chosen as the SDVF.

- **Data:** Performance Test User Cost Model
- **Function Type:** Linear Preference Function

- **Direction of Preference:** Minimize
- **Range:** XBest = \$39,000, XWorst = \$122,000
- **SDVF** $V(X_i)$**:**

$$V(X_i) = \frac{X_i - 122000}{39000 - 122000} \tag{3.5}$$

Minimize System Cost For the system cost the decision maker wants to know the total cost for the architecture. The system cost will include labor for operation and maintenance of the system, the infrastructure costs, and costs to build the system.

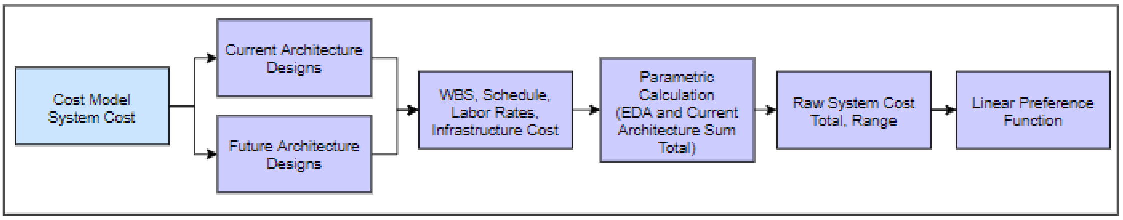

Cost Model System Cost
Current Architecture Designs
Future Architecture Designs
WBS, Schedule, Labor Rates, Infrastructure Cost
Parametric Calculation (EDA and Current Architecture Sum Total)
Raw System Cost Total, Range
Linear Preference Function

Figure 3.15: Calculation Overview for System Cost

The data systems for the executive dashboard will be used to calculate the costs of each architecture. The cost will take into account the length of time it takes to build the system, the labor rates for the teams, and cloud computing costs. The cost model will include WBS, infrastructure costs, and a development schedule for each architecture. Based on the possible system cost scores a range will be determined. The decision maker's preferences for the system cost scores was consistent over the range of values, so a linear preference function as chosen as the SDVF.

- **Data:** System Total Cost Model
- **Function Type:** Linear Preference Function
- **Direction of Preference:** Minimize

- **Range:** XBest = \$793,204, XWorst = \$16,000,000
- **SDVF** $V(X_i)$**:**

$$V(X_i) = \frac{X_i - 16000000}{793204 - 16000000} \tag{3.6}$$

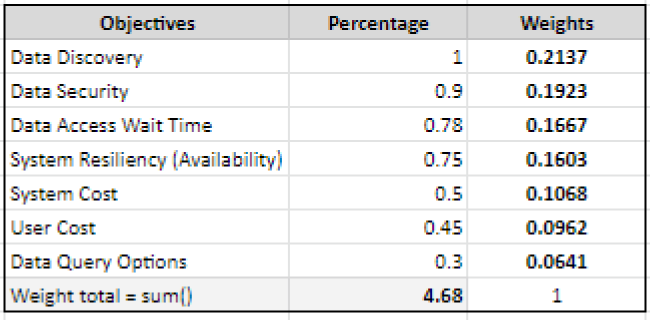
Weights The team selected the swing weight method for eliciting the weights. This method has straight forward steps for the decision maker to follow and it helps the decision maker think deeply about what objective is more important than the others and by how much.

Below are the steps to eliciting weights using the swing weight method [69]. Table 3.1 shows how the weights were calculated in excel.

1. List the all the means-objectives with their worst possible values.
2. Ask the decision maker which objective would they improve to the best value. That one gets a percentage of 100.
3. Ask the decision maker which objective they would improve next by swinging it to the best value. Ask them what percentage to assign it in relation to the top objective.
4. Go through all the objectives until each of them has a percentage.
5. Sum up the total percentage weight.
6. Divide each percentage by the total percentage weight to get the normalized weight for each objective.

Data Discovery ranked the highest for the decision maker because if data can't be discovered then it can't be used. Data Security was the next highest ranked because protecting data is a high priority to the organization. Data Access Wait Time and System Resiliency were ranked very close, but Data Access Wait Time ranked slightly higher because improving data access for users was very important to the stakeholders. System Cost was slightly more important than User Cost because it will be a larger amount of money which mattered

Table 3.1: Swing Weights



Objectives	Percentage	Weights
Data Discovery	1	**0.2137**
Data Security	0.9	**0.1923**
Data Access Wait Time	0.78	**0.1667**
System Resiliency (Availability)	0.75	**0.1603**
System Cost	0.5	**0.1068**
User Cost	0.45	**0.0962**
Data Query Options	0.3	**0.0641**
Weight total = sum()	**4.68**	1

more to the decision maker. Data Query Option was ranked last because it was viewed as a beneficial option for users but not as important as the other objectives.

3.3.3 Current Architecture (MBSE Model)

The current architecture is made up of several separate data systems that are disconnected. Each data system was built and is being maintained by a different team. Each data system has a different architecture, security, and purpose. The MBSE model is the first attempt at documenting all these systems under one architecture. It will be made up of several diagrams and elements that represent different views of the current architecture. The diagrams listed are ones that are needed for this analysis effort only. More diagrams can explain in greater detail how the systems are built and behave, but because it isn't relevant to this analysis so they are not included.

Figure 3.16 is a picture of the Model Tree. The model is comprised off all the diagrams and elements. Different views of the architecture can be created to show different aspects of the model. The tree provides a methods for organizing and accessing the different diagrams and elements.

Data Collection The EDA team collected data by conducting interviews with each of the data systems teams and reviewed the documentation they provided. Data was collected to understand each of their architecture designs and components. This data was used to create the current data architecture block definition diagram (BDD) in figure 3.17.

Performance Tests Data Requirements When collecting data for the current architecture it was important to keep focused on the data and information needed to measure it's performance. For each data system information was collected on the following measurement criteria.

- **Availability Data** Each data system team was asked about how much downtime they had have over the whole year. They were asked to provide estimates in hours.

- **Vulnerability Data** Data system teams were asked a series of questions on the security of their system. Based on their answers they were given a rating on how well their system performed for the question. Their rating data is what is used for the performance test.

- **Data Query Data** Each data system was asked about what options they provided for querying their data and if they allowed for any modifications to the query format.

- **Data Access Wait Time Data** The data system teams were asked about their process for getting access to data in their system. Each team was then asked to provide the min, max, and most likely access time. The most likely access time was used for the averages calculations.

- **Data Discoverability Data** The data systems were asked if they had any contact information or description of their data system posted anywhere a data user would be able to find. They were also asked to estimate how many people in the organization would know to contact them about their data.

- **User Cost** Labor rates and hours were collected from the Executive Dashboard Team.

- **System Cost** Each data system team was asked to provide cost information about their system including labor O&M costs and infrastructure costs.

Current Architecture BDD The Current Architecture is comprised of multiple data systems each with their own architecture. Each data system is built with different components, stores different data, and is managed by a different team. The current architecture also includes a data science environment which provides a secure place with analysis tools for data to be exported to so it can be examined by data scientists, data engineers, analysts etc. The dashboard system is the other external system used for data processing and visualization. It can store copies of data that users upload for the dashboards and can query systems for data if their is an established connection. Figure 3.17 is the block definition diagram (BDD) that documents each of the data systems in the current architecture.

The following list provides a description of each of the systems in the current architecture.

- **Finance Data System** The finance data system is the primary finance system for the organization. It stores information about the budget, invoices, and spend plan. The data system owner is concerned with who can see data from this system since it contains organization sensitive information. Because of that this data system is hard to get access to and it was not designed to support open access or high demand. Because it is a critical system it was designed with good security and enough capacity to support its current users.

- **Project Management Data System** The project management data system provides the organization a tool to host information about projects. It contains project documentation, schedules, project status, and historical documentation. The data system owner wants as many people and project teams to be using the system as possible. They market the system to the organization and are quick to approve access. They designed the system so data could be extracted in multiple ways and support demand for their user-base, but they were not concerned about security when the built the system.

- **HR Data System** The HR data system is the primary source of information about the organizations employees. It stores personally identifiable information (PII), payroll, performance evaluations, organization structure, and other human resource activities. The data system owner is protective of the data because it stores PII and only provides access after vetting each person. The system was designed to support a high demand and with good security. Every one in the organization is aware of the system an can get exports of some data after the PII information has been removed.

- **Facilities Data System** The facilities data system stores information about the organizations buildings including: seating plans, storage locations, parking capacity, inspection certifications, utilities, and other facility information. The data system owner is mainly focused on supporting their immediate stakeholders and does not advertise the system well or have an established access process. The system was built a several years ago to support only its current stakeholders. The system has not been updated in a long time. Its security might need to be updated and it was not designed for data science use.

- **Project Tracking Data System** The project tracking system is the newest system deigned to support task management in projects. It contains more granular information about what tasks people are working on. People could be working several projects and needed a tool to help manage their time. The data system owner is new and is working to establish the system more by providing quick access and beginning to market the capability in the organization. The tool was built quickly with a focus on functionality and not security or capacity.

- **Dashboard System** The Dashboard system has been around for a while and is used mainly for displaying manually uploaded data via excel.

- **Data Science Environment** The data science environment provides a secure environment data users can export data to or query data from. The environment allows data in so it can be analyzed or manipulated with data science tools, but data sources that have been combined to form a new data set have no place to be stored long term outside the environment.

Current Architecture Executive Dashboard OV-1 Figure 3.18 is an OV-1 diagram showing how the current architecture supports the executive dashboard project. The data user who represents a member of the executive dashboard team has to directly contact each data owner in order to get access to the data systems. The data user has to manually upload and download data from data systems to the data science environment and then to the dashboard system. Not all the data systems allow direct connections to the data science environment or the dashboard system, so there is a lot of manual and time consuming work that a data user need to do.

Current Architecture Physical Design Figure 3.19 is a diagram showing the physical implementation of the data systems. The diagrams shows how each data systems utilizes different cloud services as their enabling infrastructure. The data from this diagram will play a role when estimating monthly costs for each of the data systems in the System Cost Model section 3.3.7. The enabling infrastructure could be swapped out for a different cloud provider or purchased hardware.

Note: AWS services were used because it was easy to represent both the current and future architectures and calculate costs. In real-world organizations that infrastructure will likely have more variety in the data systems infrastructure. This was done as a simplification for the example.

Use Case Data Collection The essential use cases derived from the Executive Dashboard stakeholders in section 3.3.2 were used as a discussion point for collecting data from the data system owner and their team. The data from the discussion was used to understand how each of the systems would be utilized and how well they would perform for the Executive Dashboard project. The use case diagram 3.20 shows the general use cases for the whole current architecture. All the data systems can be generalized to these use cases. Each use case has a behavior diagram that describes how the user and data systems interact.

Activity Diagrams Figures 3.21 and 3.22 are activity diagrams that describe the general workflows for each of the data systems.

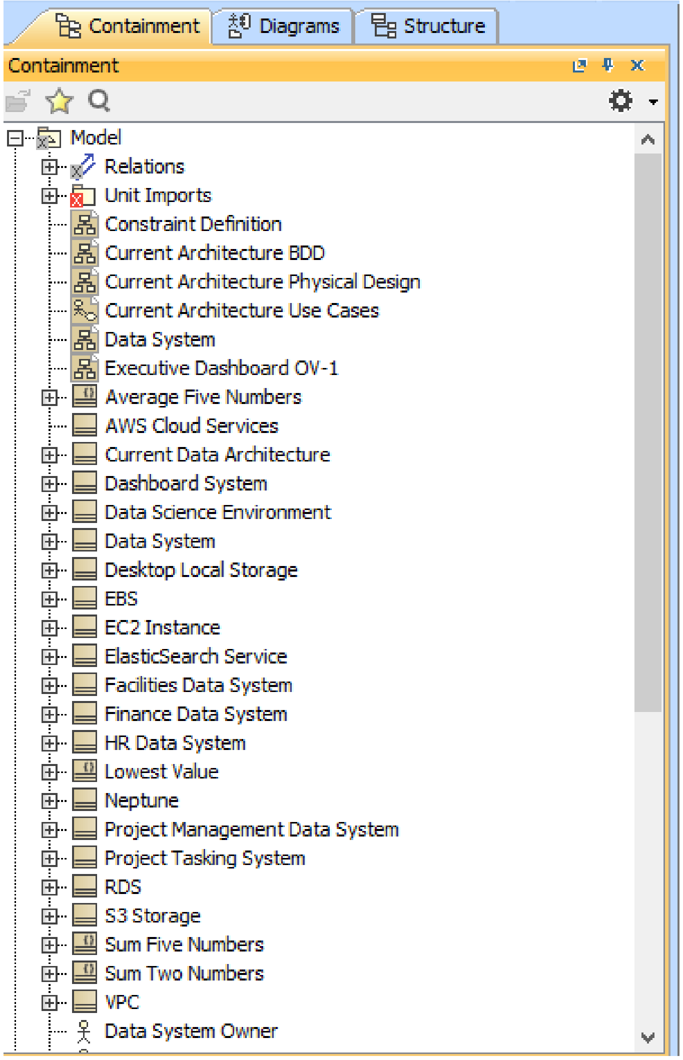
The Discover Data Activity diagram describes how a data user would find data in the current architecture. The user would perform a search first to see if they could find information about data systems that could have the data they are interested in. They would look for information about the data system, contact information for the team that runs the data system, how to get access to the data, if there is an API and other people or projects who may use the data. The search will either continue until actionable information is found and they can request access to the data or they pursue the second method of discovering data. If the search is unsuccessful the data user will start asking people if they have heard of a data system that may contain the information they are looking for. Every person they ask there is a chance they might know about a source of data. This probability is based on the population being all the people who work in the organization and the subset of people who have actionable information needed to get access to the data needed. Once all data has been found the Data Discovery process stops.

The request access process for each data system has been generalized to this activity diagram. Once the data user has information on how to contact the Data System owner they send an email request. The data owner might not see the email right away for a variety of reasons. Once they review the request they make a decision to approve or reject it. If they reject it then the data user will need to adjust their email to provide more information on why they need access to the data source. If the data owner accepts the request they notify their engineering team to provide the data user access to the system. The team will take time to work through their process for adding users. The team might not be very responsive to their emails which could also take time. The process ends once the user has received an email that they have access to the data.

Parametric Diagram Figure 3.23 and figure 3.24 are parametric diagrams for the current architecture. They are graphical representations of how the overall current architecture value parameters will be calculated based on the individual data system value parameters.

The actual calculation of the scores is done in section 3.3.6.

Each data system was evaluated using the performance tests and has their own Availability Score, Vulnerability Score, Data Query Score, Data Access Time, Data Discovery Score, and User Cost Score. The 3.23 parametric diagram is one example showing how each of the current architecture performance scores will be calculated by averaging the results of the five data systems. Each score has it's own parametric diagram that describes how the score is calculated using the averaging constraint.



```
Model
    Relations
    Unit Imports
    Constraint Definition
    Current Architecture BDD
    Current Architecture Physical Design
    Current Architecture Use Cases
    Data System
    Executive Dashboard OV-1
    Average Five Numbers
    AWS Cloud Services
    Current Data Architecture
    Dashboard System
    Data Science Environment
    Data System
    Desktop Local Storage
    EBS
    EC2 Instance
    ElasticSearch Service
    Facilities Data System
    Finance Data System
    HR Data System
    Lowest Value
    Neptune
    Project Management Data System
    Project Tasking System
    RDS
    S3 Storage
    Sum Five Numbers
    Sum Two Numbers
    VPC
    Data System Owner
```

Figure 3.16: Model Tree

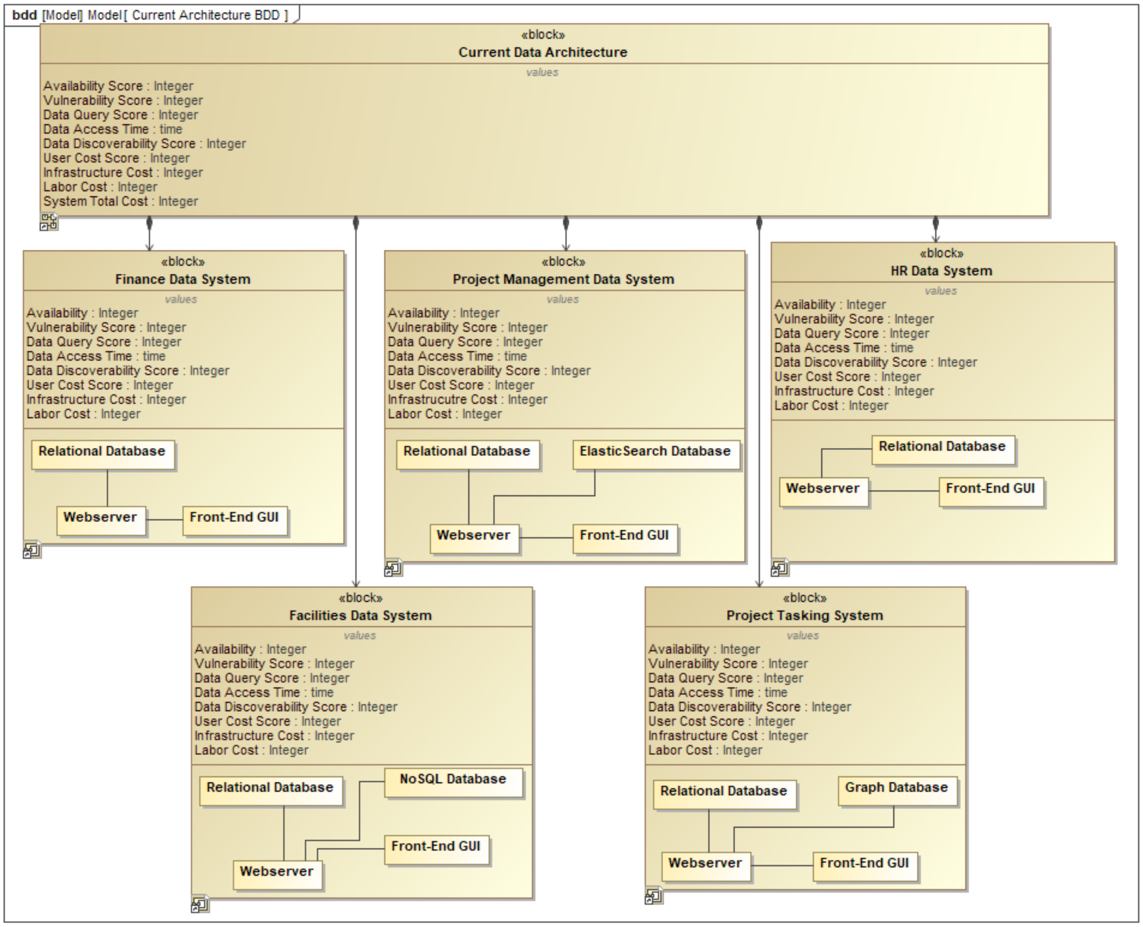

bdd [Model] Model [Current Architecture BDD]
«block»
Current Data Architecture
values
Availability Score : Integer
Vulnerability Score : Integer
Data Query Score : Integer
Data Access Time : time
Data Discoverability Score : Integer
User Cost Score : Integer
Infrastructure Cost : Integer
Labor Cost : Integer
System Total Cost : Integer
«block»
Finance Data System
values
Availability : Integer
Vulnerability Score : Integer
Data Query Score : Integer
Data Access Time : time
Data Discoverability Score : Integer
User Cost Score : Integer
Infrastructure Cost : Integer
Labor Cost : Integer
Relational Database
Webserver
Front-End GUI
«block»
Project Management Data System
values
Availability : Integer
Vulnerability Score : Integer
Data Query Score : Integer
Data Access Time : time
Data Discoverability Score : Integer
User Cost Score : Integer
Infrastrucutre Cost : Integer
Labor Cost : Integer
Relational Database
ElasticSearch Database
Webserver
Front-End GUI
«block»
HR Data System
values
Availability : Integer
Vulnerability Score : Integer
Data Query Score : Integer
Data Access Time : time
Data Discoverability Score : Integer
User Cost Score : Integer
Infrastructure Cost : Integer
Labor Cost : Integer
Relational Database
Webserver
Front-End GUI
«block»
Facilities Data System
values
Availability : Integer
Vulnerability Score : Integer
Data Query Score : Integer
Data Access Time : time
Data Discoverability Score : Integer
User Cost Score : Integer
Infrastructure Cost : Integer
Labor Cost : Integer
Relational Database
NoSQL Database
Front-End GUI
Webserver
«block»
Project Tasking System
values
Availability : Integer
Vulnerability Score : Integer
Data Query Score : Integer
Data Access Time : time
Data Discoverability Score : Integer
User Cost Score : Integer
Infrastructure Cost : Integer
Labor Cost : Integer
Relational Database
Graph Database
Webserver
Front-End GUI

Figure 3.17: Current Architecture BDD

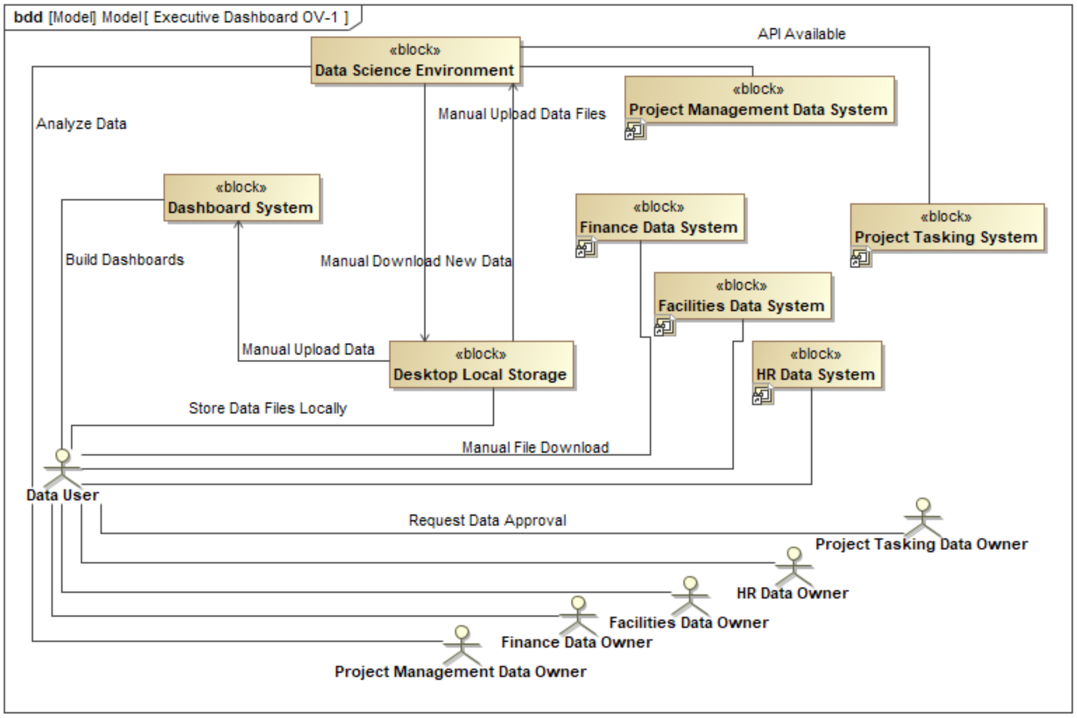


Figure 3.18: Current Executive Dashboard OV-1

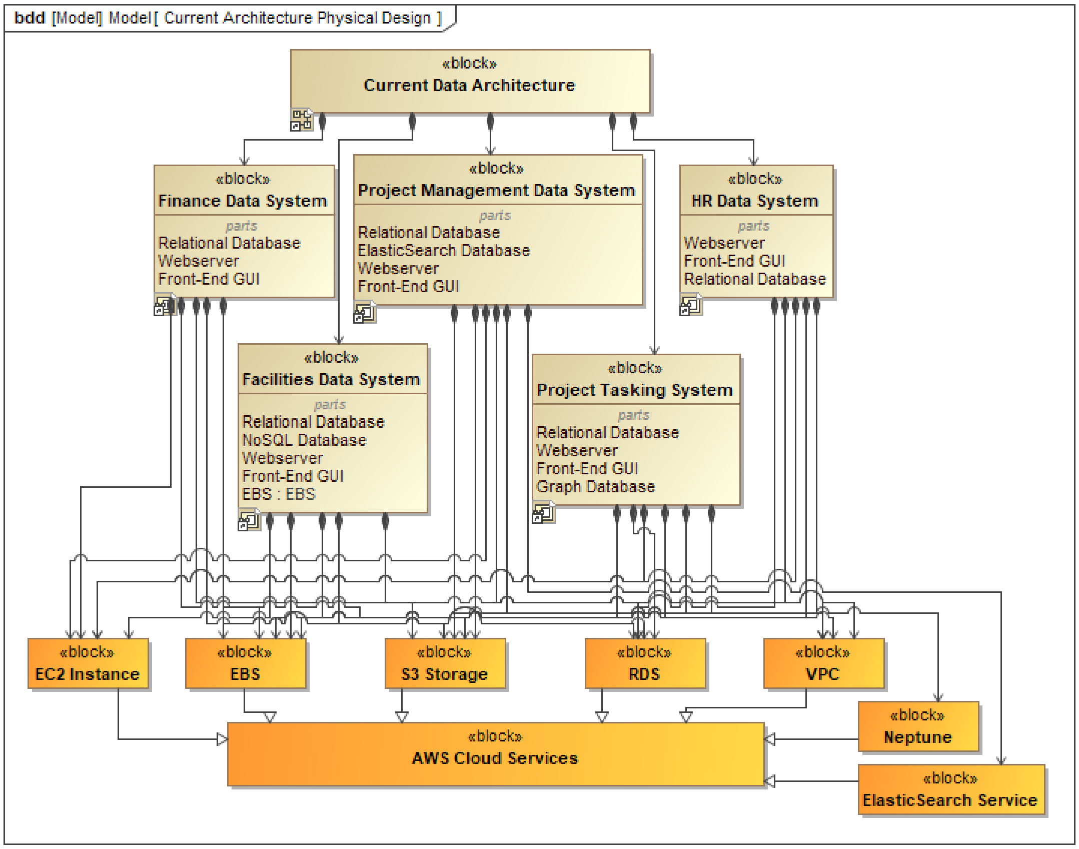


Figure 3.19: Current Architecture Physical Design

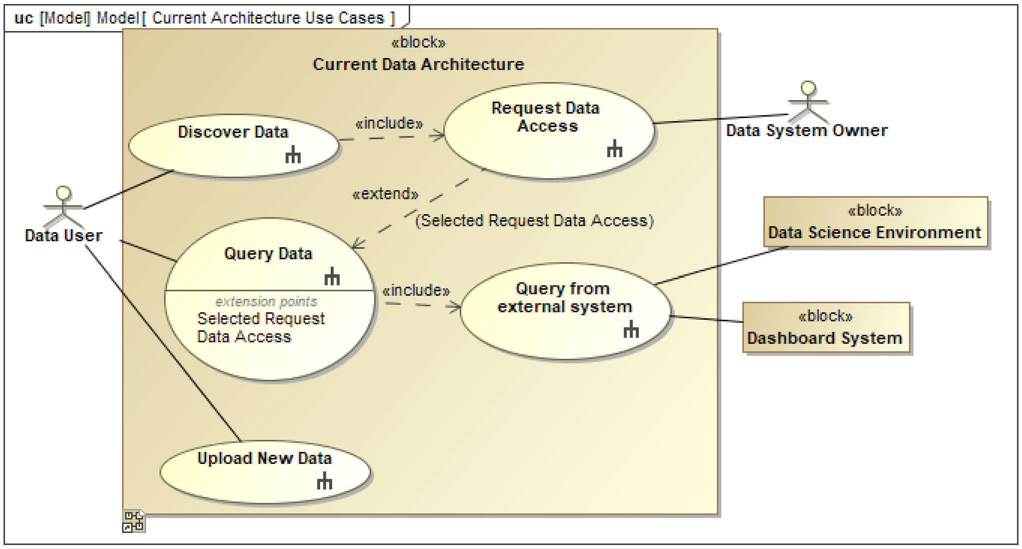


Figure 3.20: Current Data Architecture Use Cases

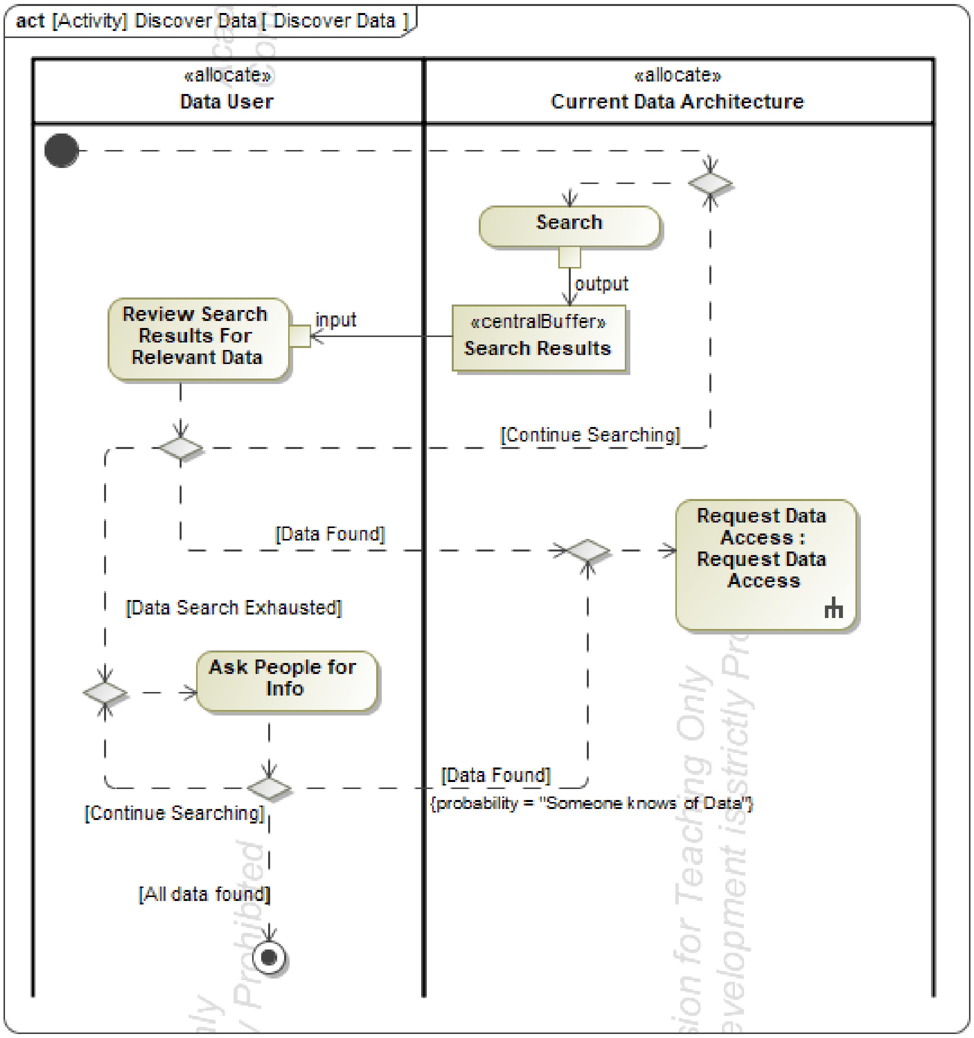


Figure 3.21: Current Architecture Discover Data Activity Diagram

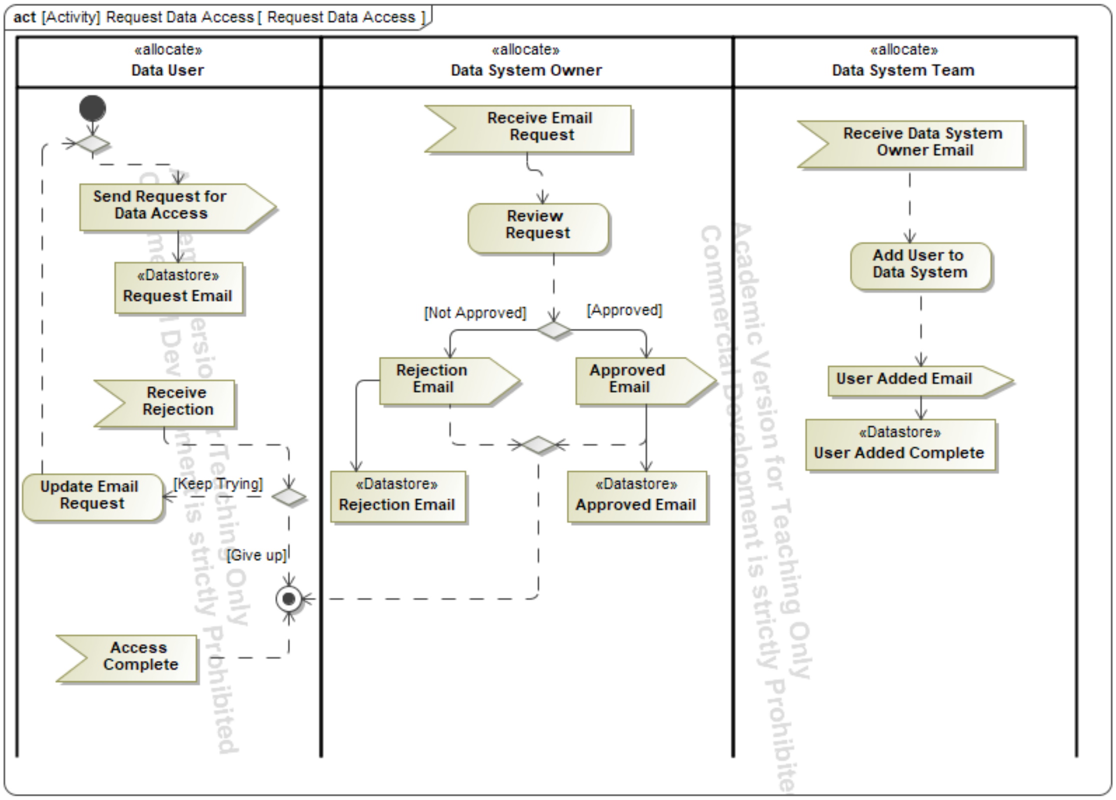


Figure 3.22: Current Architecture Request Access Activity Diagram

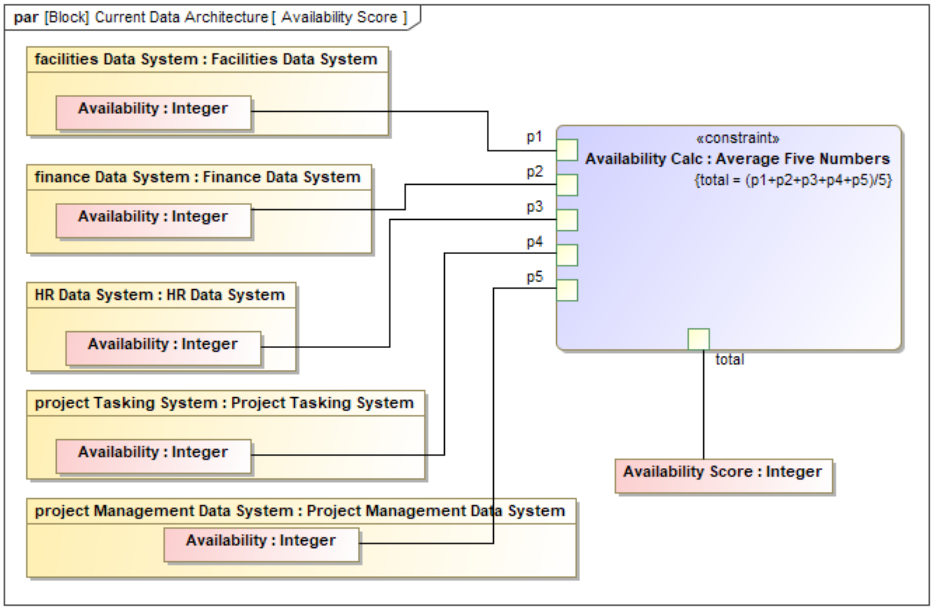

par [Block] Current Data Architecture [Availability Score]
facilities Data System : Facilities Data System
Availability : Integer
finance Data System : Finance Data System
Availability : Integer
HR Data System : HR Data System
Availability : Integer
project Tasking System : Project Tasking System
Availability : Integer
project Management Data System : Project Management Data System
Availability : Integer
p1
p2
p3
p4
p5
«constraint»
Availability Calc : Average Five Numbers
{total = (p1+p2+p3+p4+p5)/5}
total
Availability Score : Integer

Figure 3.23: Current Availability Parametric Diagram

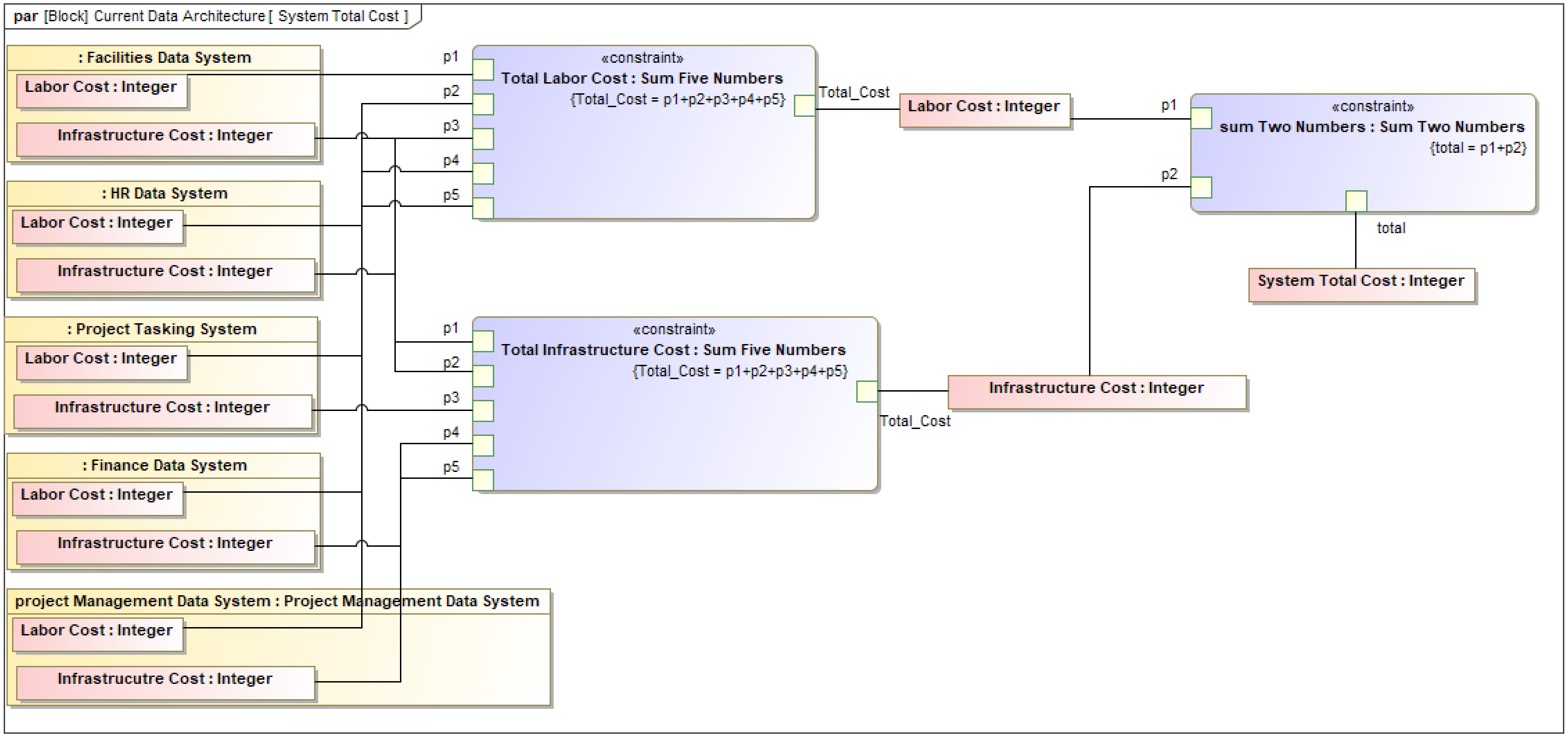

par [Block] Current Data Architecture [System Total Cost]
: Facilities Data System
Labor Cost : Integer
Infrastructure Cost : Integer
: HR Data System
Labor Cost : Integer
Infrastructure Cost : Integer
: Project Tasking System
Labor Cost : Integer
Infrastructure Cost : Integer
: Finance Data System
Labor Cost : Integer
Infrastructure Cost : Integer
project Management Data System : Project Management Data System
Labor Cost : Integer
Infrastrucutre Cost : Integer
p1
p2
p3
p4
p5
«constraint»
Total Labor Cost : Sum Five Numbers
{Total_Cost = p1+p2+p3+p4+p5}
Total_Cost
p1
p2
p3
p4
p5
«constraint»
Total Infrastructure Cost : Sum Five Numbers
{Total_Cost = p1+p2+p3+p4+p5}
Total_Cost
Labor Cost : Integer
Infrastructure Cost : Integer
p1
p2
«constraint»
sum Two Numbers : Sum Two Numbers
{total = p1+p2}
total
System Total Cost : Integer

Figure 3.24: Current Architecture Total System Cost Parametric Diagram

To calculate the current architecture's total cost, each data system will first calculate their labor and infrastructure costs. The labor costs and infrastructure costs across all five data systems is then summed up to the the total labor costs and total infrastructure costs. The final cost of the current architecture is calculated with both those values.

3.3.4 Future EDA (MBSE Model)

The future enterprise data architecture will be a single system with subsystem components. The MBSE model is a design of what the potential future system could be. There are diagrams that describe the functional structure and physical implementation of the design. There are diagrams that describe the behavior of the system. The behavior diagrams show how external systems and users will interact with the system. The designs are easily modified, so if the decision maker wants to add or remove a feature the design can easily be modified and all the diagrams will be updated. The EDA design can have more detail or less detail, for this analysis effort the EDA model only contains the necessary detail to perform the decision analysis.

Figure 3.25 is a picture of the EDA model tree. Because the EDA model could have several design iterations it was necessary to group parts of the design together and organize the model. Figure 3.26 is called a package diagram. It provides a view of how the model is organized.

Data Collection Data for the EDA design came from a variety of sources. Interviews with the data system teams, the executive dashboard team, the security team and the decision maker. Information on the stakeholder's desires for the future system and their concerns were taken into account. Research was conducted on new data architecture technologies and methods for implementation. Data from current EDA technologies were used to develop the design.

EDA Block Definition Diagram Figure 3.27 is the block definition diagram (BDD) describing the five subsystem components that make up the EDA architecture. It is a view

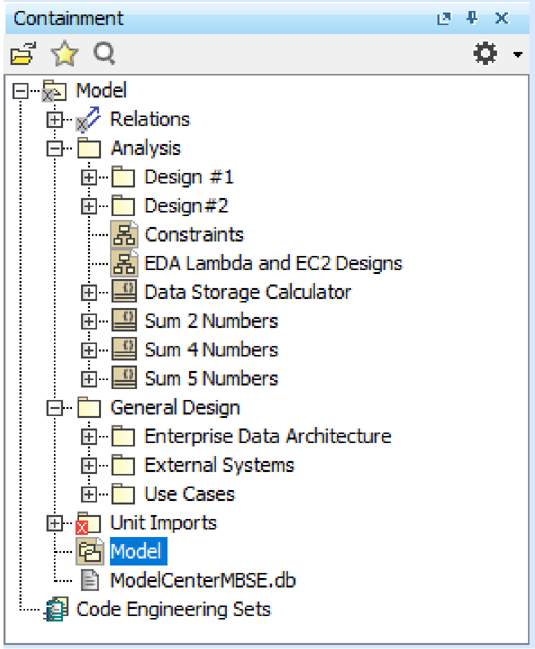


Figure 3.25: EDA Model Tree

of the functional subsystems EDA needs to operate. Below is a description of each of the subsystems and their functions.

- **Request Processing System** Is the subsystem that handles all external and some internal request of the EDA system. It hosts all the external facing end-points for queries to be submitted through, then determines what action needs to be taken.

- **Website** The website is the primary interface data users will use to discover and request data. It will also be used to enforce roles and access by the data owners and managers. It has two major component systems that make up the majority of the functionality.

  Data Catalog The data catalog component system is a data registry that holds information about all data sources. It contains descriptions that data users would want to see and provides a request mechanism for users to easily request access to

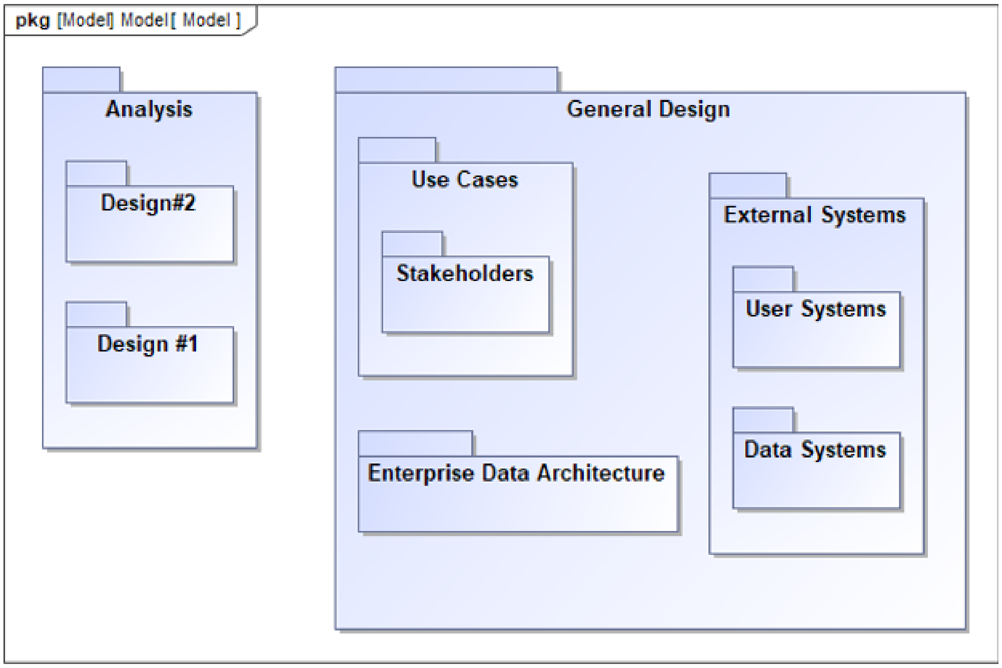


Figure 3.26: EDA Package Diagram

the data. The data catalog allows for new sources of data to be registered with the EDA system. Data owner can manage who has access to their data and easily add or remove data users.

Account Management The account management component system is a user registry that holds information about users of the system. It stores the relationships of who users need permission from to get access to the system and data. The account management specifies the types of roles users can have in the EDA system.

- **Security** The security subsystem manages the encryption keys and enforces access to data and the system. It maintains the list of which users have access to which data.
- **Data Processing** The data processing system computes the data queries. It pulls data from the different data storage options and performs the query needed. It keeps the data sources up to date with frequent data pulls. It maintains the data pipelines

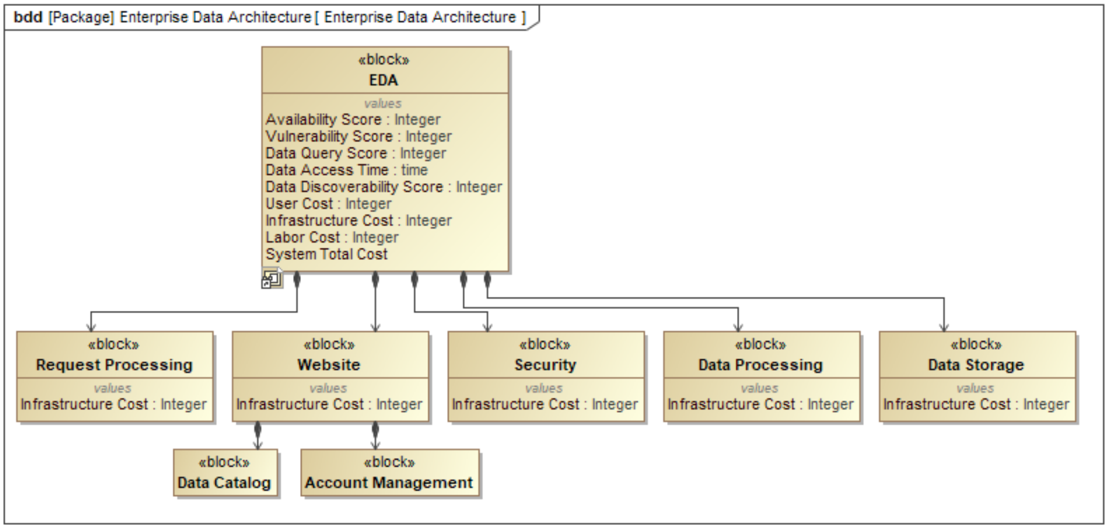

bdd [Package] Enterprise Data Architecture [Enterprise Data Architecture]
«block»
EDA
values
Availability Score : Integer
Vulnerability Score : Integer
Data Query Score : Integer
Data Access Time : time
Data Discoverability Score : Integer
User Cost : Integer
Infrastructure Cost : Integer
Labor Cost : Integer
System Total Cost
«block»
Request Processing
values
Infrastructure Cost : Integer
«block»
Website
values
Infrastructure Cost : Integer
«block»
Security
values
Infrastructure Cost : Integer
«block»
Data Processing
values
Infrastructure Cost : Integer
«block»
Data Storage
values
Infrastructure Cost : Integer
«block»
Data Catalog
«block»
Account Management

Figure 3.27: Future EDA BDD

and ETL transactions.

- **Data Storage** The data storage subsystem stores the data and keeps it organized and secure.

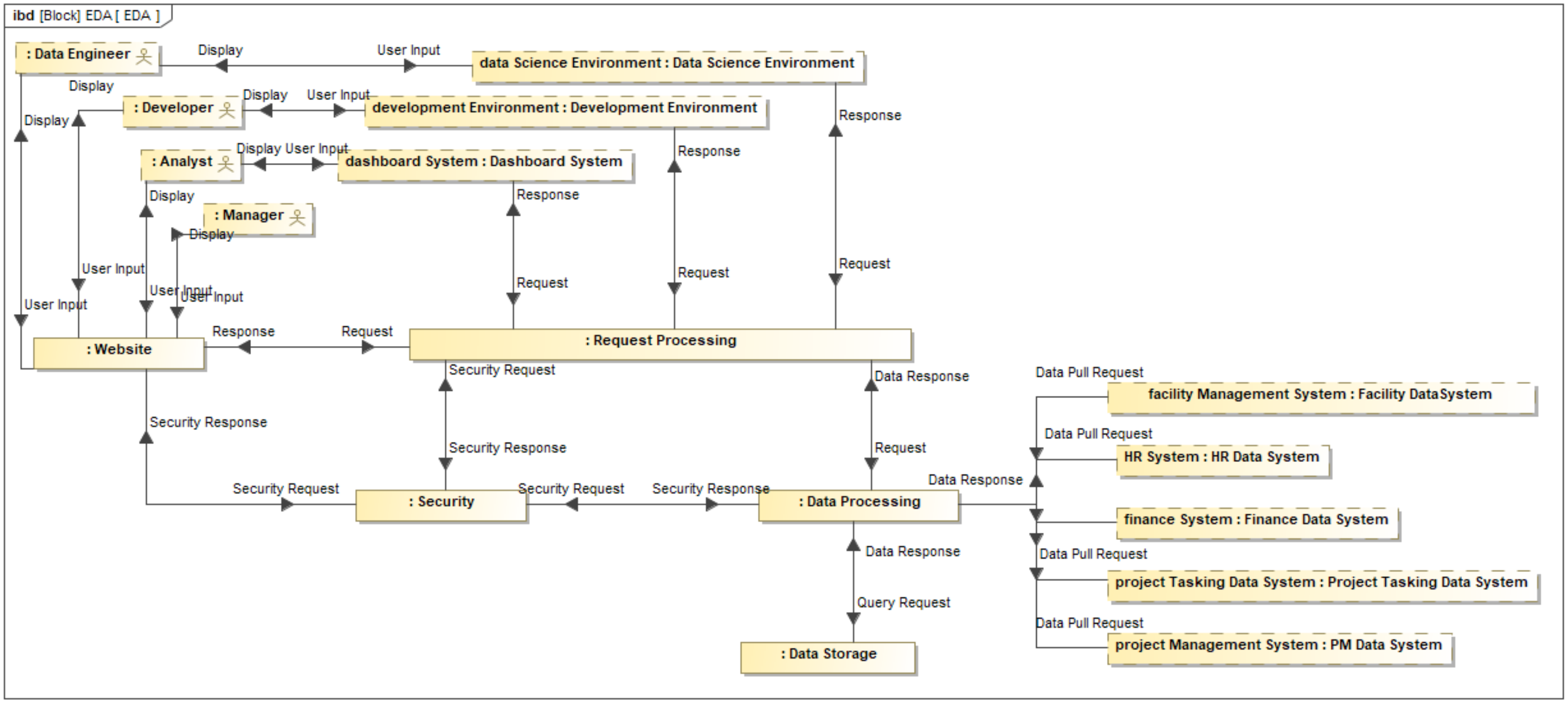


Figure 3.28: Future EDA IBD

EDA Internal Block Diagram The internal block diagram (IBD) describes how the subsystems communicate. Figure 3.28 shows how each subsystem interacts with each other and the external systems and users. Different data users provide input to the website or they perform queries to the system via their analytic environments. The request processing system and website are the only subsystems that receive and process external requests. The security subsystem communicates with the website, request processing, and data processing system to provide security reviews of requests. The data processing system ingests data from the data systems, processes it, and puts the data into storage.

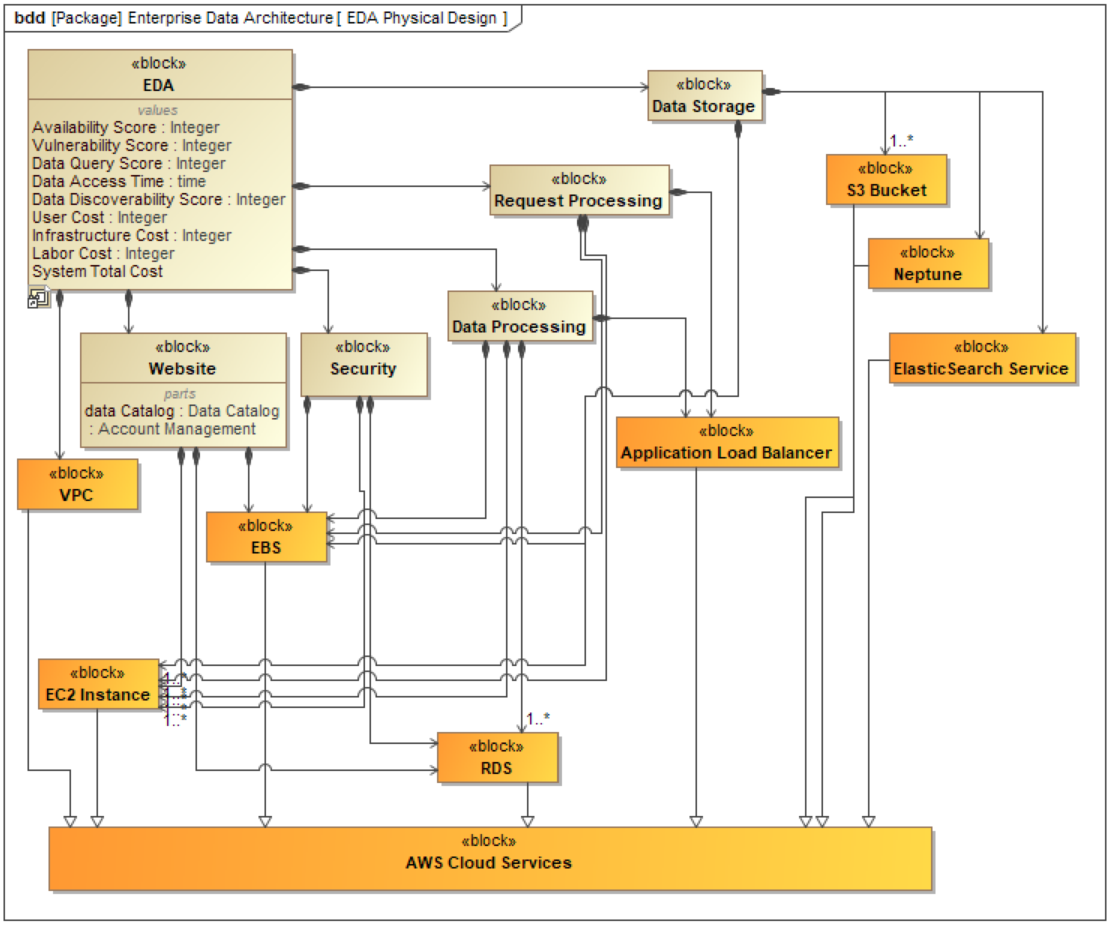

bdd [Package] Enterprise Data Architecture [EDA Physical Design]
«block» EDA
values
Availability Score : Integer
Vulnerability Score : Integer
Data Query Score : Integer
Data Access Time : time
Data Discoverability Score : Integer
User Cost : Integer
Infrastructure Cost : Integer
Labor Cost : Integer
System Total Cost
«block» Data Storage
«block» Request Processing
«block» Data Processing
«block» Website
parts
data Catalog : Data Catalog
: Account Management
«block» Security
«block» S3 Bucket
1..*
«block» Neptune
«block» ElasticSearch Service
«block» Application Load Balancer
«block» VPC
«block» EBS
«block» EC2 Instance
1..*
«block» RDS
1..*
«block» AWS Cloud Services

Figure 3.29: Future EDA Physical Architecture

EDA Physical Implementation There are many options for implementing an enterprise data architecture. Commercial tools can be used, servers can be bought, or different cloud providers and services can be used. The EDA Development Initiative Team selected an AWS cloud implementation as their physical implementation of the EDA because their organization had experience building cloud architectures with AWS. Figure 3.29 shows how each subsystem will use different AWS services to build out that subsystem. This physical architecture diagram will be used when calculating the infrastructure costs in the system cost model section 3.3.7.

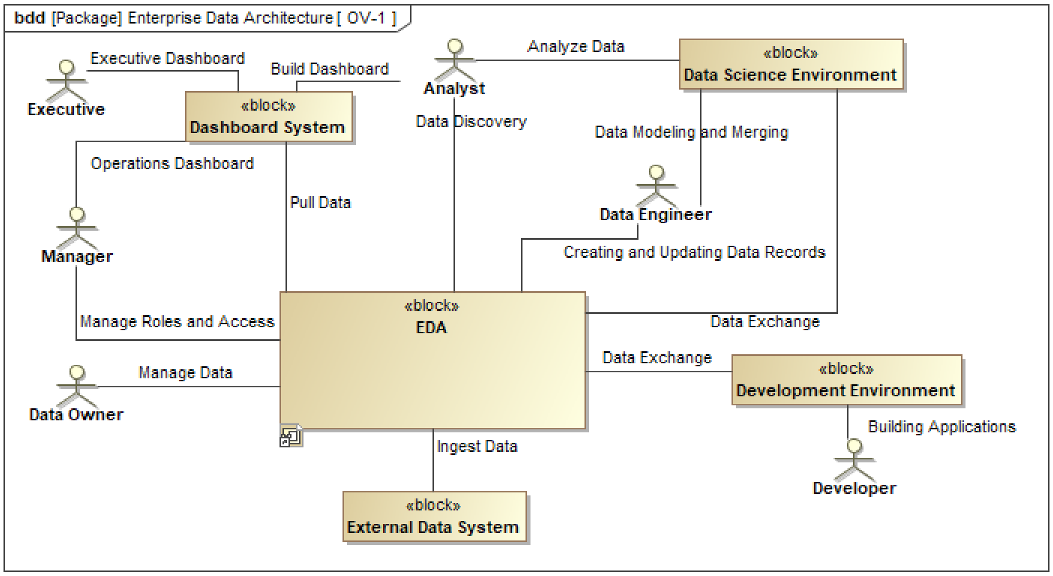


Figure 3.30: Future EDA OV-1

EDA OV-1 Figure 3.30 describes what general usage operations of the EDA system would look like for different stakeholders. Analysts, Data Engineers, and Developers will be using EDA to discover data and export it to their analytic systems for further analysis depending on their projects. Managers will be approving access to the system and data. The original data systems will be feeding data to the EDA. These systems may be decommissioned if the organization finds that their functionality isn't useful anymore, but their data will still be stored in the EDA. For the intermediate step in upgrading the architecture most of the data systems will still exist. Eventually those applications may be modified to use EDA as their primary data storage and decommission their data storage components.

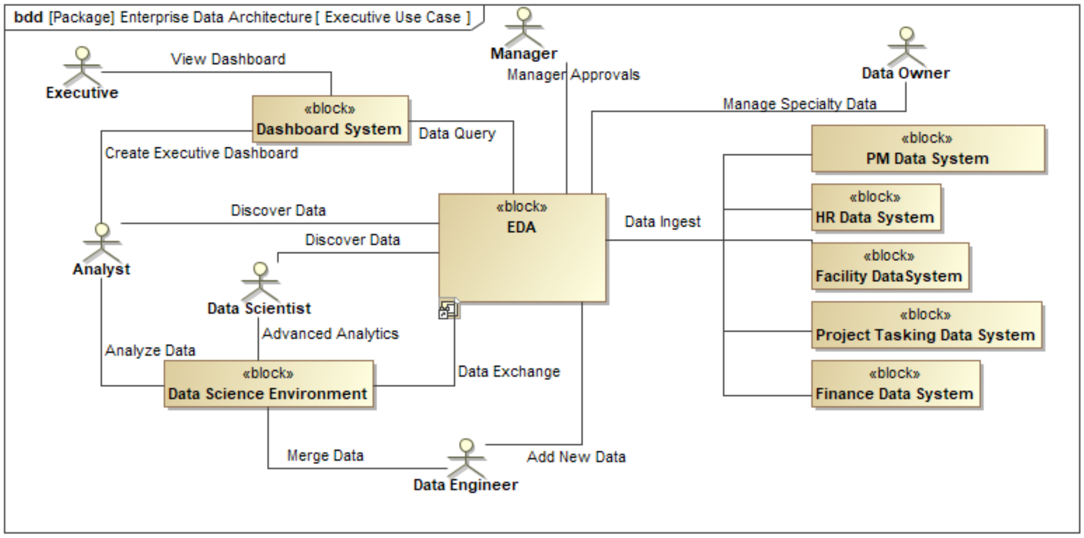


Figure 3.31: Future EDA Executive Dashboard OV-1

Figure 3.31 describes how the Executive Dashboard team would use the EDA system for their data science project. All team members would use EDA to discover data and query data from their analytic environments. New data could be registered with the EDA system making it easier for the dashboard to be built. Managers and Data owners approve data access requests and access to the EDA system itself. The data systems ingest data into the EDA system frequently so the data in EDA is up to date. The Dashboard is able to connect to the EDA system and pull the data needed for the executive dashboard.

Use Cases The enterprise data architecture has the same essential use cases from figure 3.3.2 and some additional ones in order to support the new business processes the EDA system creates. The EDA system will need managers to approve access for data users of the system. Each data user they will need to create an account with the EDA system

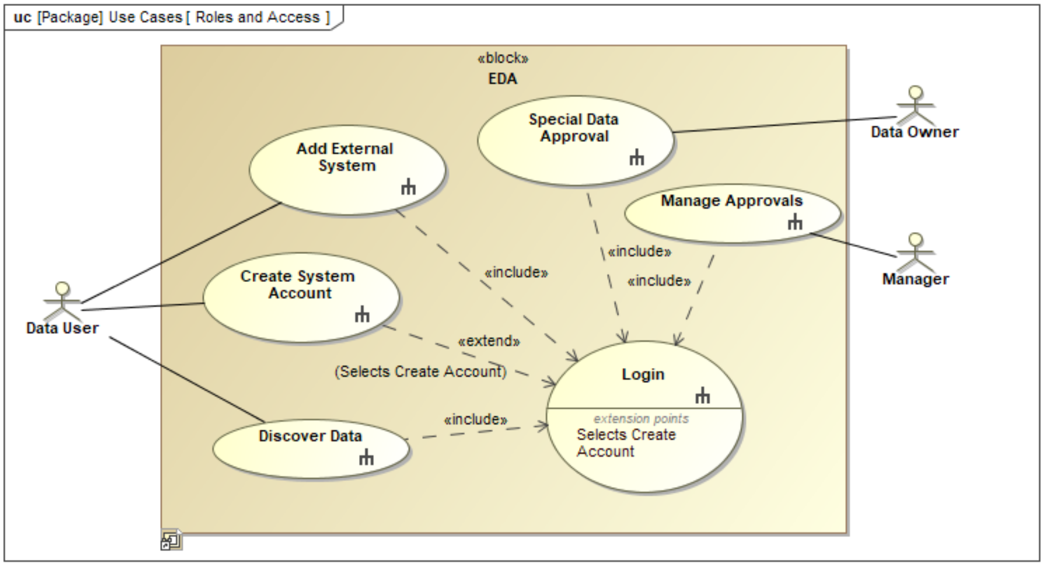


Figure 3.32: Future EDA Use Case Diagram

and register their external system accounts. The external system accounts are the analytic environments that have been approved for data export such as the dashboard system and data science environment. Another new use case will be logging into the EDA system, which is included in several of the other use cases.

Figure 3.33 describe three high-level use cases. The Data Ingest use case is how data systems will interact with EDA in the future. The Data Query use case is how data will be extracted from EDA in the future to the approved external user systems. The New Data Record Use Case addresses the issue of data users not having a process or enabling infrastructure to put new data sources. A data user will now be able to register a new data source that can be used by other data users in the organization. Each of these use cases are described with either an Activity, Sequence, or State-Machine Diagram.

Activity, Sequence, and State-Machine Diagram The following diagrams are used to describe behavior of the future system. Each of the diagrams was derived from one of the use cases or is a decomposed activity from another diagram.

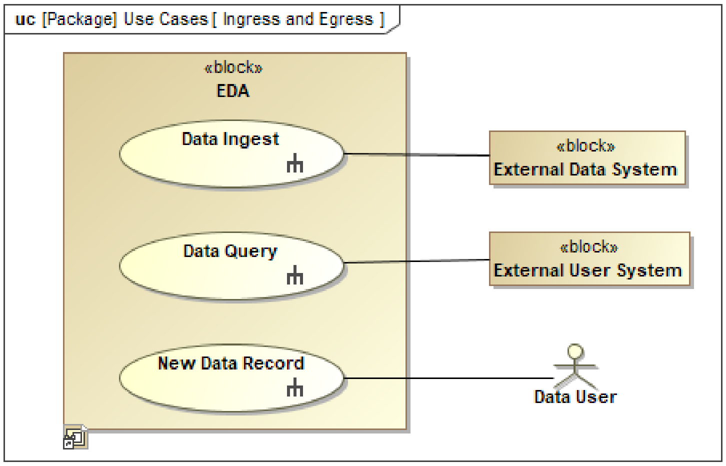


Figure 3.33: Future EDA Use Case Diagram Ingress and Egress

Data Discovery Activity Diagram Figure 3.34 is an activity diagram describing the data discovery process using the EDA system. A data user will interface with the system via the Website subsystem, which will allow the user to search all the data sets listed in the data catalog. Once a data source is found the user can request access and is notified that their request is being processed.

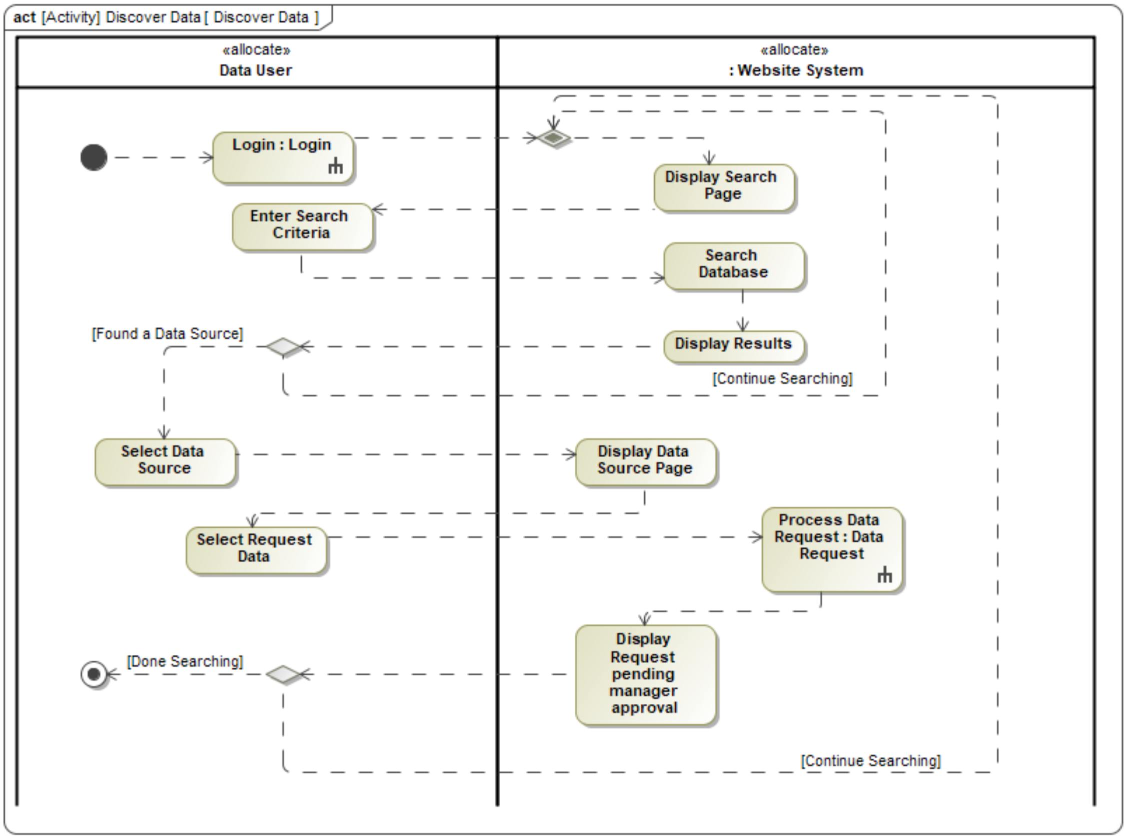


Figure 3.34: Future EDA Data Discovery Activity Diagram

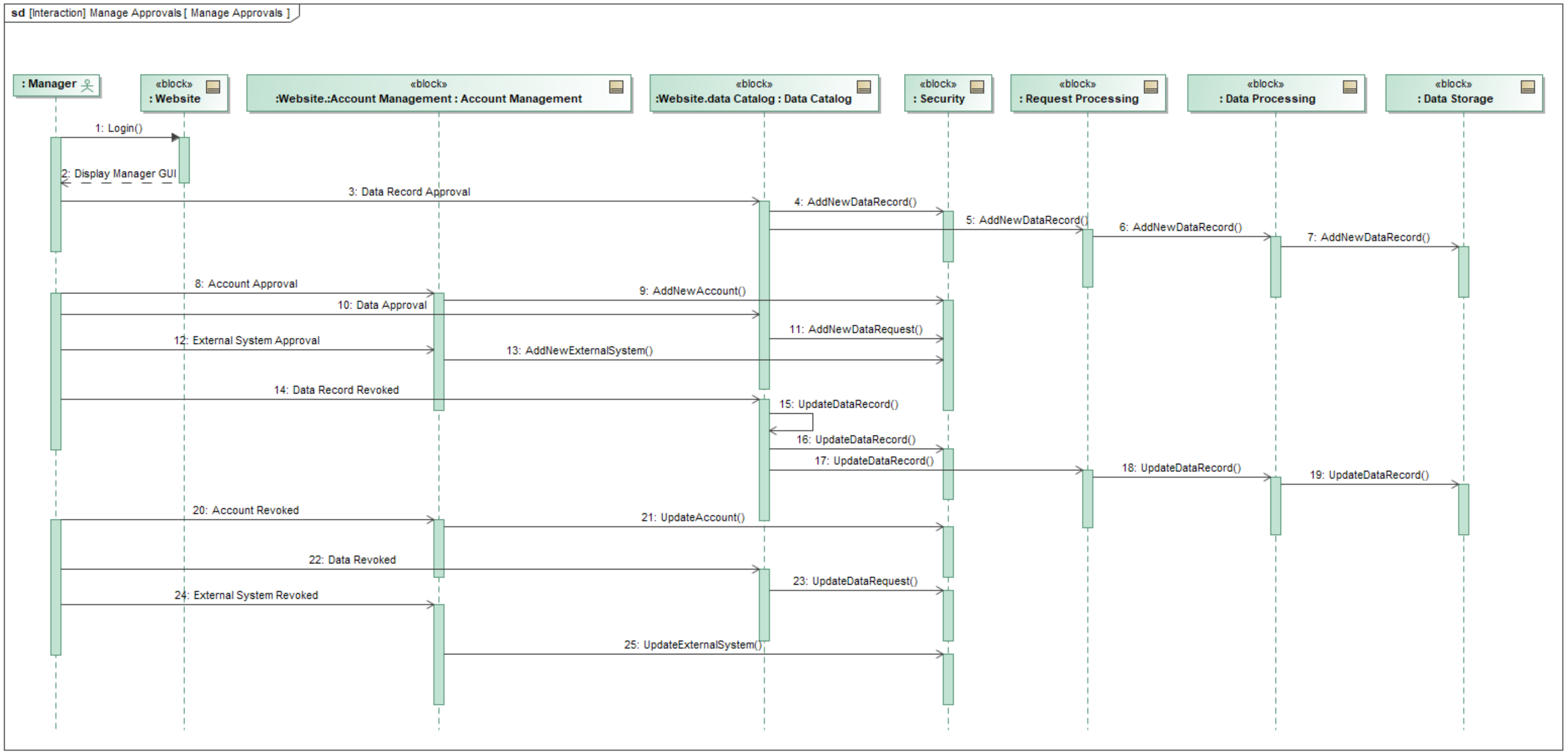


Figure 3.35: Future EDA Manager Approval Sequence

Manager Approval Sequence Diagram Figure 3.35 is a sequence diagram that describes the manager's approval process and how that information is then communicated to other components of the system. The Manager is able to approve and revoke a person's access permissions to the system or data source. The manager also approved new data be registered with the system, which can also be revoked. This diagram shows the different operations each of the subsystems and components will need in order to support the use case functionality.

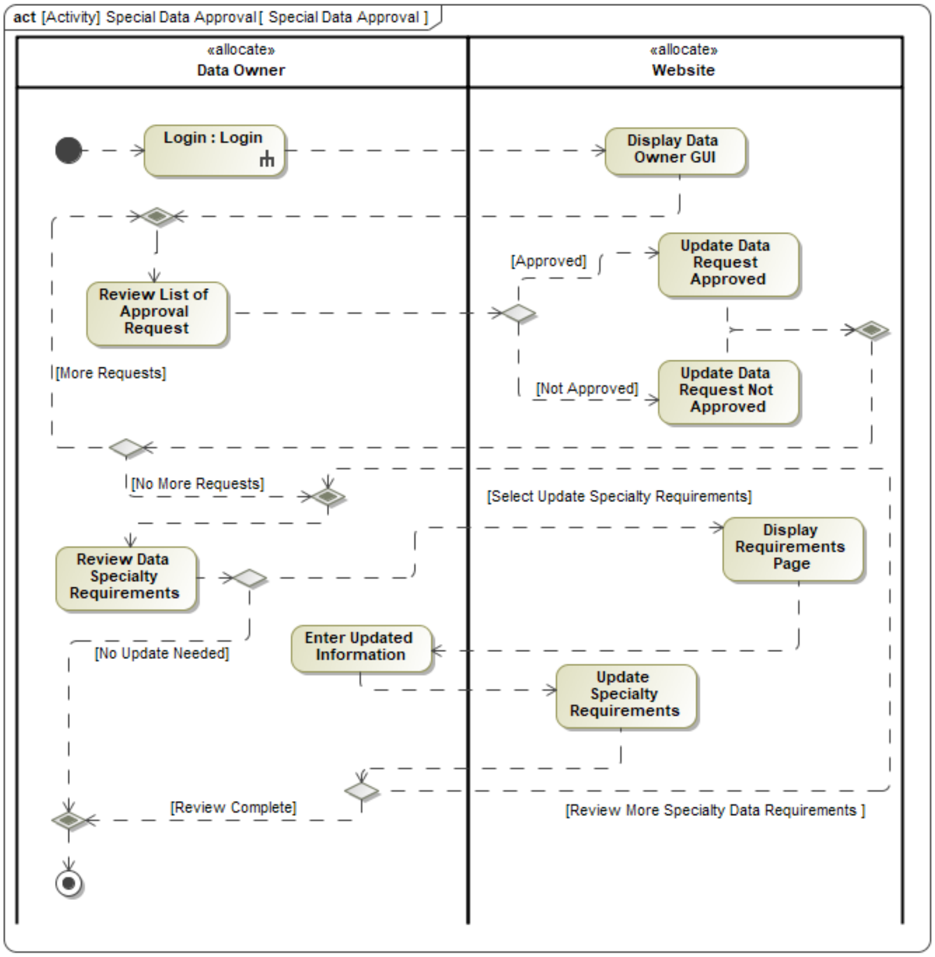

act [Activity] Special Data Approval[Special Data Approval]
«allocate»
Data Owner
«allocate»
Website
Login : Login
Display Data Owner GUI
[Approved]
Update Data Request Approved
Review List of Approval Request
[More Requests]
[Not Approved]
Update Data Request Not Approved
[No More Requests]
[Select Update Specialty Requirements]
Display Requirements Page
Review Data Specialty Requirements
[No Update Needed]
Enter Updated Information
Update Specialty Requirements
[Review Complete]
[Review More Specialty Data Requirements]

Figure 3.36: Future EDA Special Data Access

Data Owner Approval Activity Diagram Figure 3.36 is an activity diagram representing the special case of data needing Data Owner approval. The default permission model will be for a manger to approve their employees access to data, but there are some data sources that need an extra level of control because they contain sensitive information. This diagram follows the same general flow as the manager approval, except it is represented in an activity diagram instead of a sequence diagram. The Data Owner also has the option to modify their data permission requirements in the system.

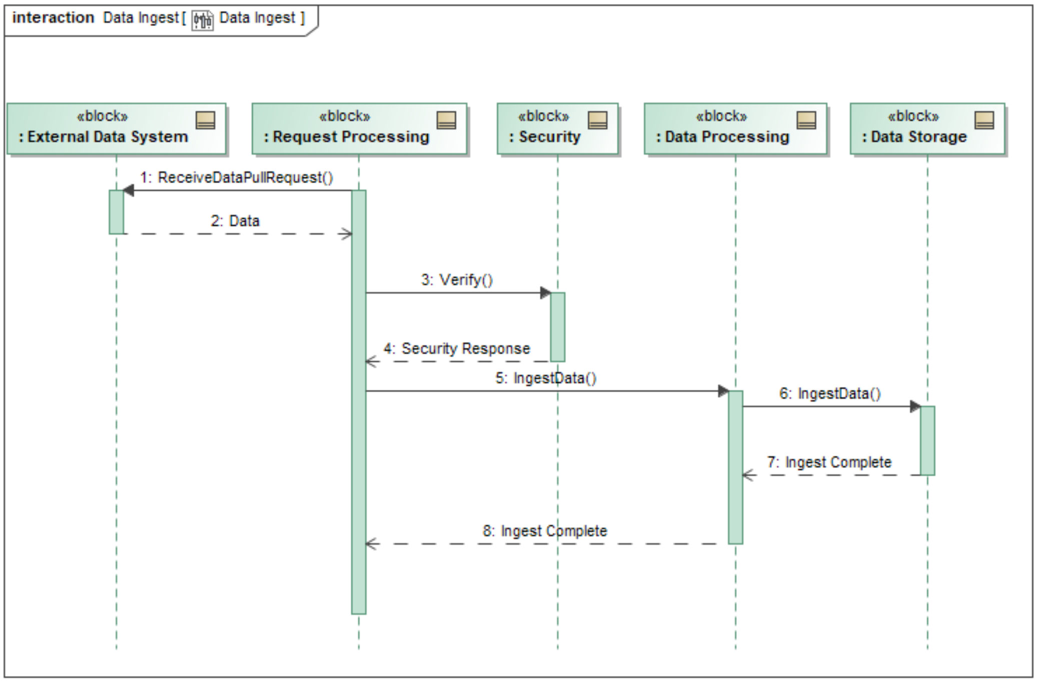

interaction Data Ingest [Data Ingest]
«block» : External Data System
«block» : Request Processing
«block» : Security
«block» : Data Processing
«block» : Data Storage
1: ReceiveDataPullRequest()
2: Data
3: Verify()
4: Security Response
5: IngestData()
6: IngestData()
7: Ingest Complete
8: Ingest Complete

Figure 3.37: Future EDA Data Ingest

Data Ingest Sequence Diagram Figure 3.37 is a sequence diagram showing how data is ingested into EDA. It shows each of the operations the subsystems will need and what order they occur in for the ingest functionality. The request processing system initiates the data pull and verifies the response with the security system before passing the data to the data processing system to extract, transform, and load (ETL) the data and store it in the data storage system.

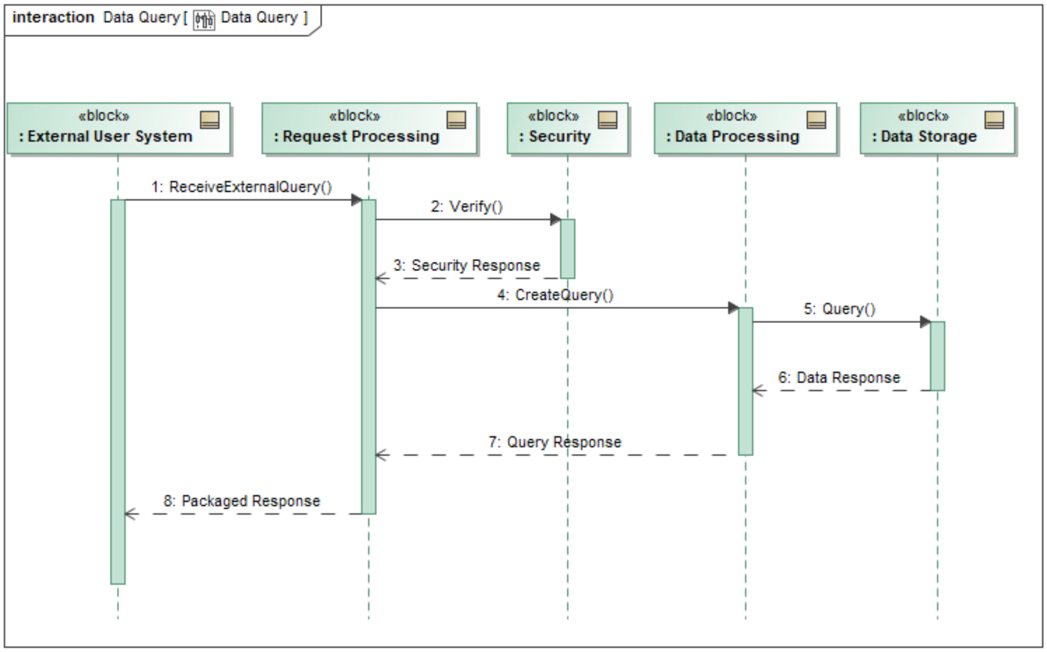


Figure 3.38: Future EDA Data Query

Data Query Sequence Diagram Figure 3.38 is a sequence diagram showing how data can be queried from the enterprise data architecture. The behavior is generalize to be from an external data system, which could be either the dashboarding system or data science environment. The request processing system verifies the request with the security system before initiating the query with the data processing system. The data processing system performs the query and sends it back to the request processing system which repackages the data in a secure response format to be sent to the external system.

Account, Data Request, and External System Request State-Machine Diagrams Figure 3.39 has three state-machine diagrams that describe the same general behavior for three types of objects within the system. System accounts, data requests, and external systems requests have the same states and state transitions. They enter a Pending

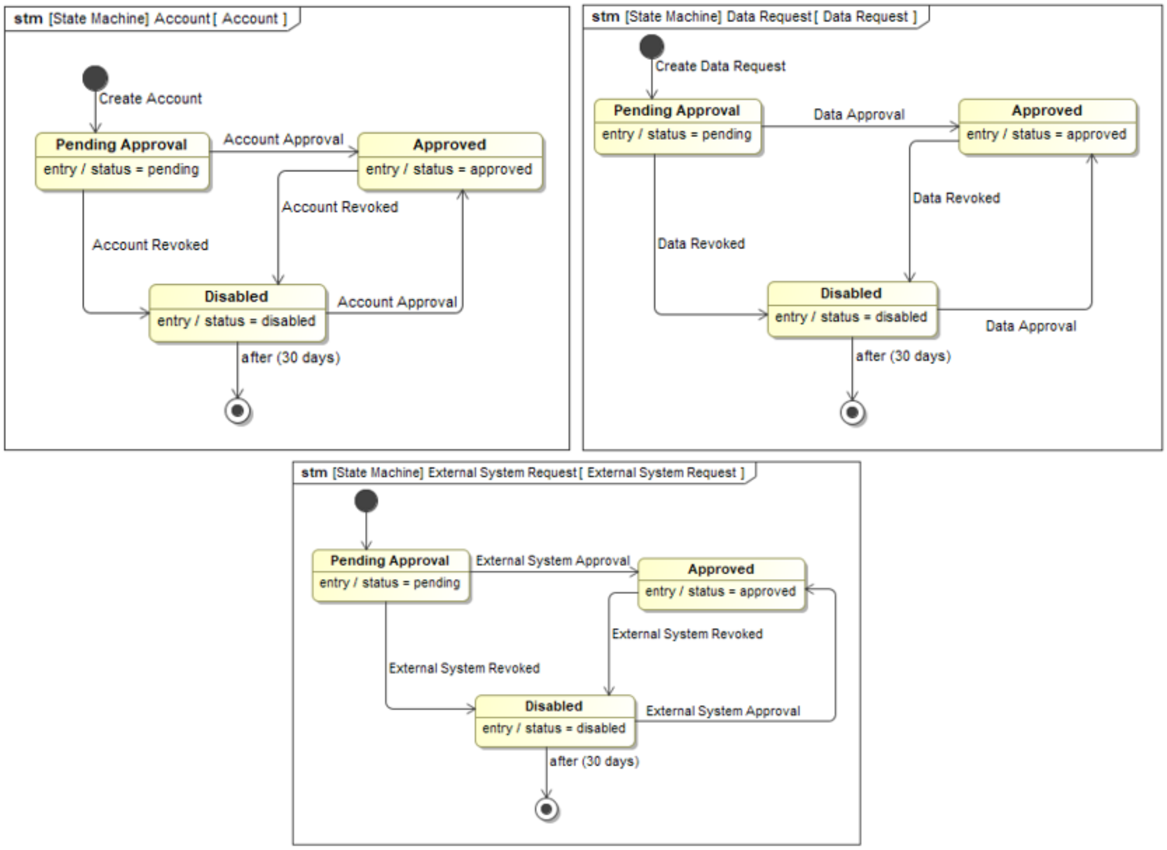


Figure 3.39: Future EDA State Machine Diagrams

Approval state after they are created. The objects are then either approved or disapproved. If the object it is disapproved, it has 30 days to transition to the approved state before the request is terminated. The account object is generated when a user requests to create an account. The external system request and the data request objects are created after a user has an account and they are registering their external system accounts and requesting access to data. Either a manger or data owner will approve or disapprove the requests.

New Data Record Sequence and State-Machine Diagrams Figure 3.40 shows two diagrams which describe the behavior registering a new data source with the EDA system. The sequence diagram shows the high-level request process a user would make

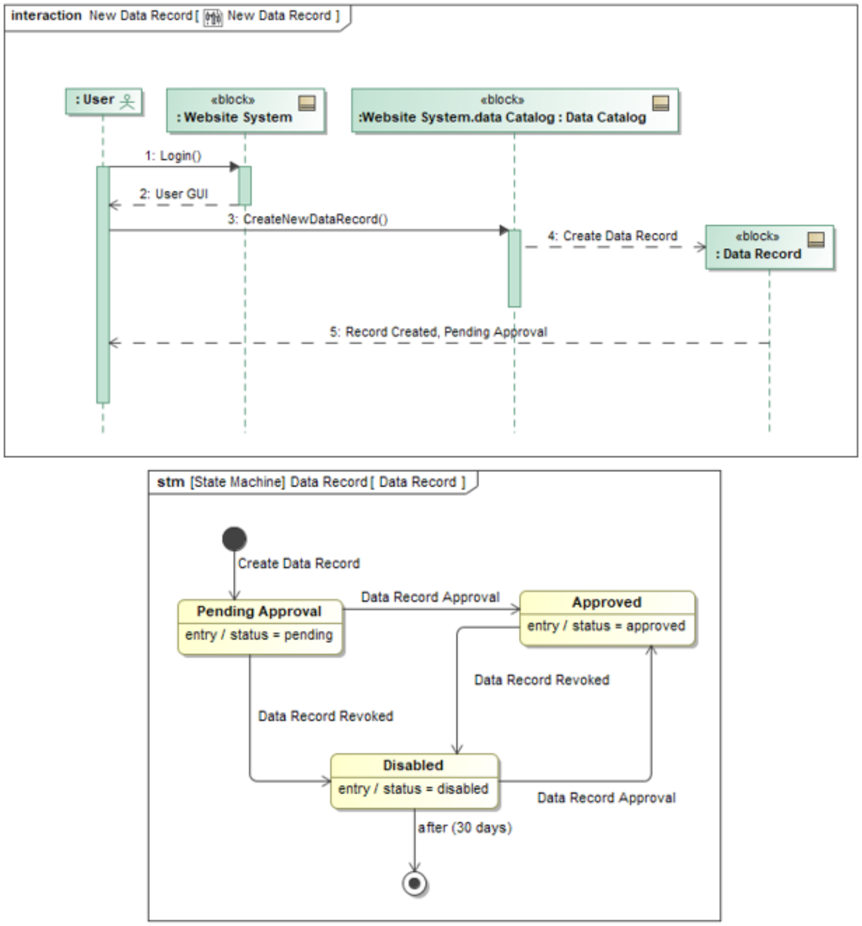


Figure 3.40: Future EDA New Data Record

when registering the data source with the system. The state-machine diagram shows the different states the data record can be in. The state-machine diagram is the same life-cycle as the previous state-machine diagrams.

Parametric Model The performance scores are calculated differently than the current architecture's performance score. Since the current architecture is comprised of separate systems, each data system needed to be evaluated and the the results of all the systems aggregated. For the EDA performance scores, the aggregation is not necessary. The EDA system will be evaluated as a whole since it is a single system. So there is not need to create a parametric diagram describing the performance scores for the EDA System.

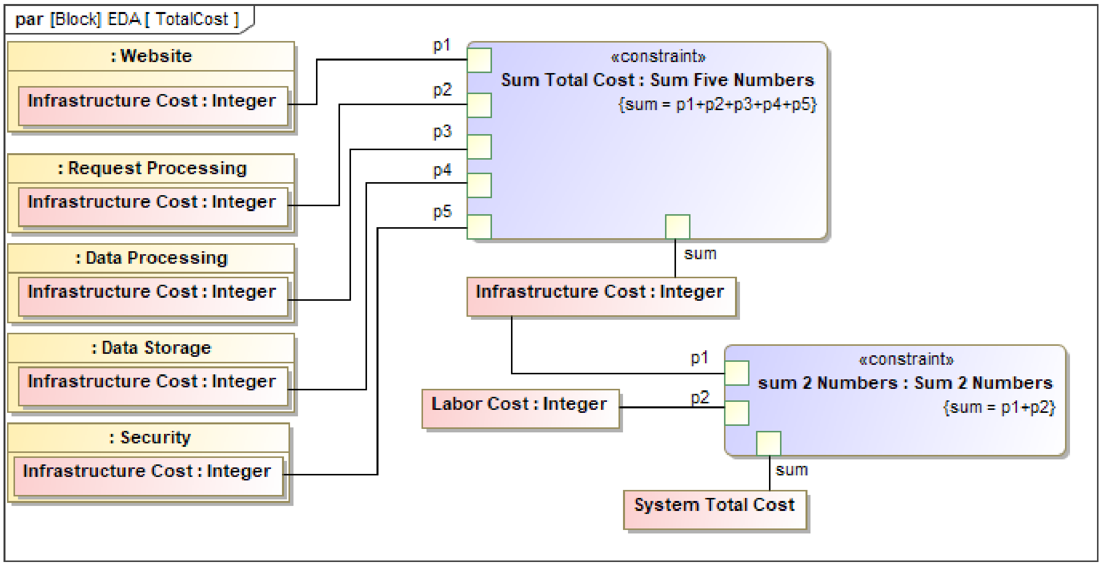

par [Block] EDA [TotalCost]
: Website
Infrastructure Cost : Integer
: Request Processing
Infrastructure Cost : Integer
: Data Processing
Infrastructure Cost : Integer
: Data Storage
Infrastructure Cost : Integer
: Security
Infrastructure Cost : Integer
p1
p2
p3
p4
p5
«constraint»
Sum Total Cost : Sum Five Numbers
{sum = p1+p2+p3+p4+p5}
sum
Infrastructure Cost : Integer
Labor Cost : Integer
p1
p2
«constraint»
sum 2 Numbers : Sum 2 Numbers
{sum = p1+p2}
sum
System Total Cost

Figure 3.41: EDA Total Cost Parametric Diagram

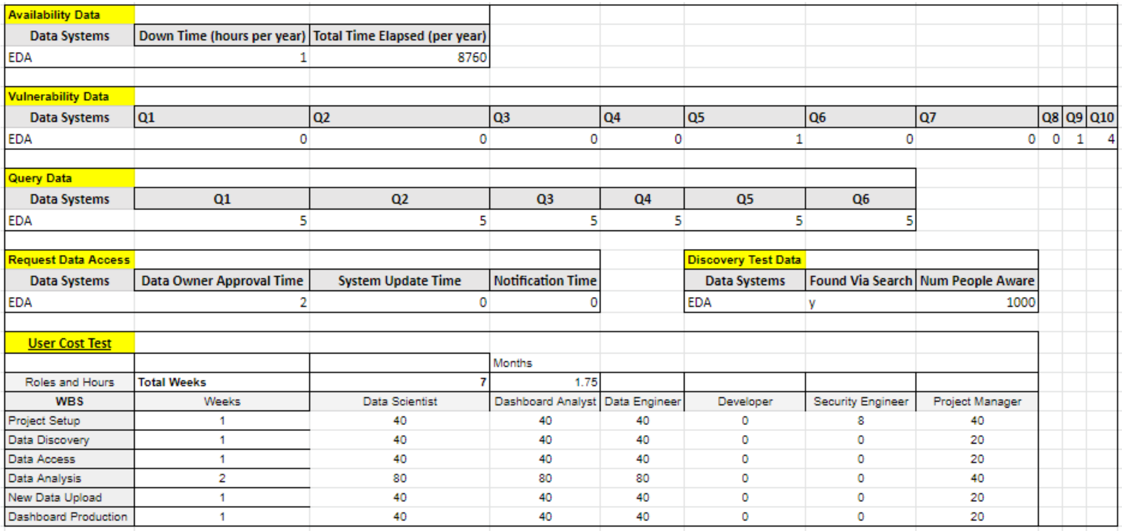
There is a need to create a parametric diagram to capture total cost. Figure 3.41 describes the EDA total cost calculation. Each subsystem will have it's own infrastructure cost, which then needs to be aggregated. The labor cost will be calculated separately as it's own value parameter. The total cost is then the combined cost of the infrastructure and labor.

Future Architecture MBSE Performance Data Summary Table 3.2 summarizes all the data that will be needed to evaluate the performance of the future enterprise data architecture. Each of the data points has been collected for each of the performance tests. These are the data that will be input into each of the performance tests in section 3.3.6.

3.3.5 Generating Alternatives

Both the current architecture and future architecture have been documented using model-based systems engineering. Discussions with the decision maker and stakeholders about how

Table 3.2: EDA Data Architecture Data Table



Availability Data		
Data Systems	Down Time (hours per year)	Total Time Elapsed (per year)
EDA	1	8760

Vulnerability Data										
Data Systems	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
EDA	0	0	0	0	1	0	0	0	1	4

Query Data						
Data Systems	Q1	Q2	Q3	Q4	Q5	Q6
EDA	5	5	5	5	5	5

Request Data Access			
Data Systems	Data Owner Approval Time	System Update Time	Notification Time
EDA	2	0	0

Discovery Test Data		
Data Systems	Found Via Search	Num People Aware
EDA	y	1000

User Cost Test							
			Months				
Roles and Hours	Total Weeks	7	1.75				
WBS	Weeks	Data Scientist	Dashboard Analyst	Data Engineer	Developer	Security Engineer	Project Manager
Project Setup	1	40	40	40	0	8	40
Data Discovery	1	40	40	40	0	0	20
Data Access	1	40	40	40	0	0	20
Data Analysis	2	80	80	80	0	0	40
New Data Upload	1	40	40	40	0	0	20
Dashboard Production	1	40	40	40	0	0	20

to go from the current status quo architecture to the future architecture are what drove the generation of alternatives. Not all the data systems can be totally replaced with the future EDA system and will still need to exist in the architecture. The decision centered around whether to improve the current architecture to address data science project needs or to build the enterprise data architecture. The decision maker wanted to know what the difference in cost and value would be between the alternative approaches. The options for implementation are the alternatives the decision maker needs to decide between. Each alternative will be evaluated by the framework to help determine which is the best solution for the organization.

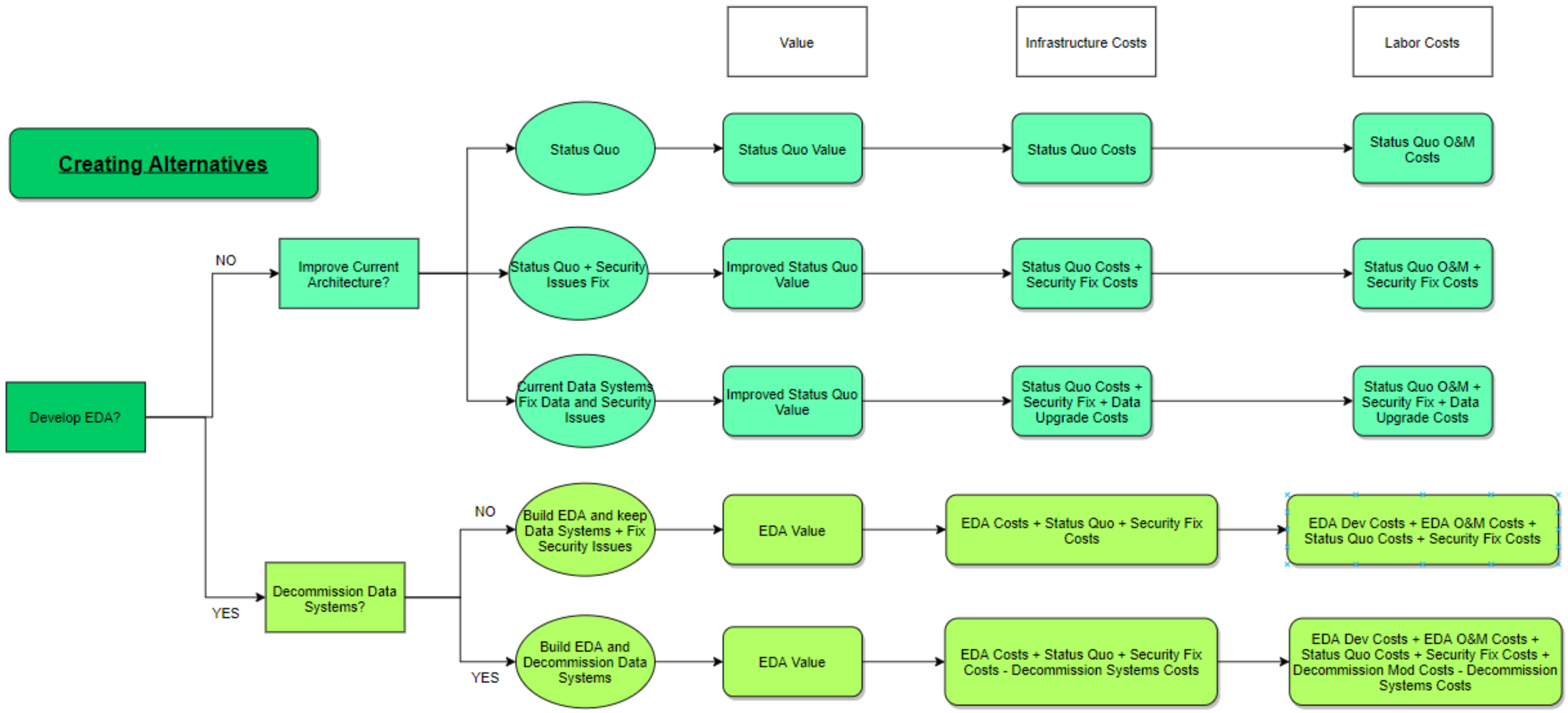
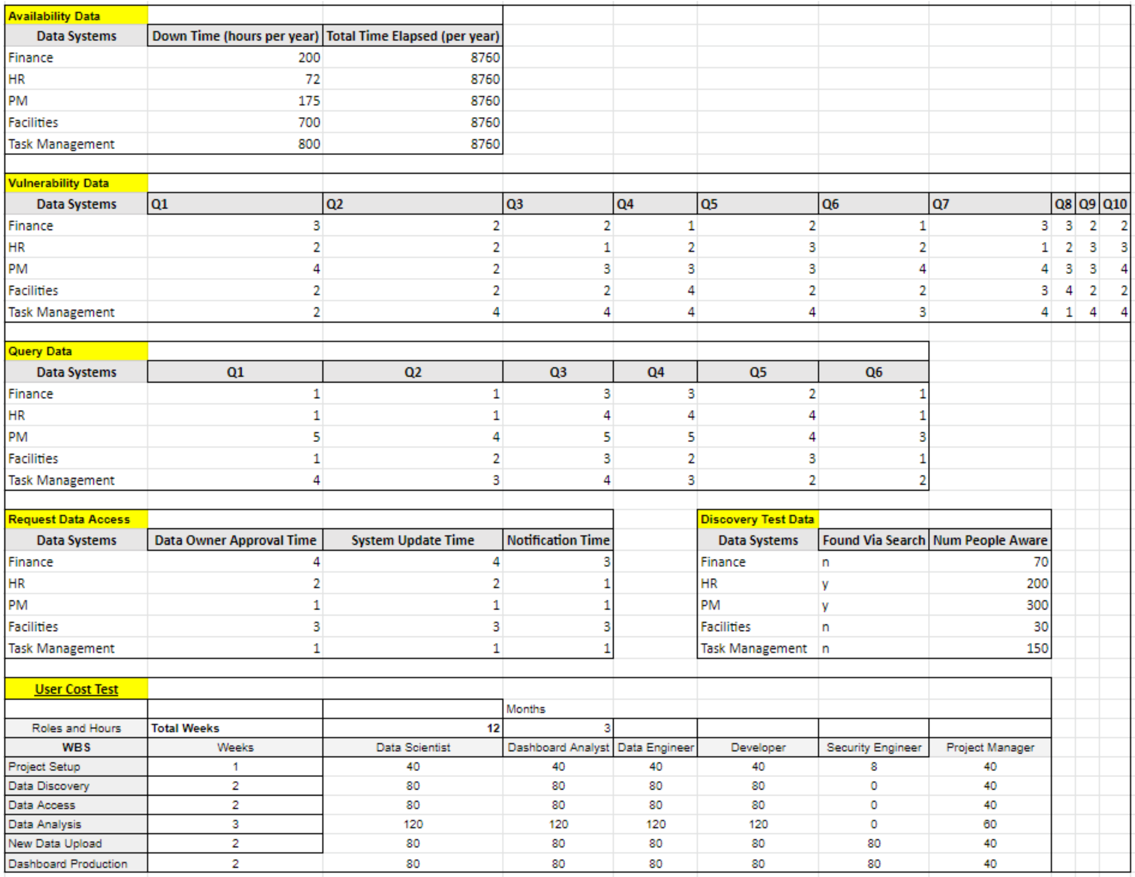

Creating Alternatives
Value
Infrastructure Costs
Labor Costs
Develop EDA?
NO
Improve Current Architecture?
Status Quo
Status Quo Value
Status Quo Costs
Status Quo O&M Costs
Status Quo + Security Issues Fix
Improved Status Quo Value
Status Quo Costs + Security Fix Costs
Status Quo O&M + Security Fix Costs
Current Data Systems Fix Data and Security Issues
Improved Status Quo Value
Status Quo Costs + Security Fix + Data Upgrade Costs
Status Quo O&M + Security Fix + Data Upgrade Costs
YES
Decommission Data Systems?
NO
Build EDA and keep Data Systems + Fix Security Issues
EDA Value
EDA Costs + Status Quo + Security Fix Costs
EDA Dev Costs + EDA O&M Costs + Status Quo Costs + Security Fix Costs
YES
Build EDA and Decommission Data Systems
EDA Value
EDA Costs + Status Quo + Security Fix Costs - Decommission Systems Costs
EDA Dev Costs + EDA O&M Costs + Status Quo Costs + Security Fix Costs + Decommission Mod Costs - Decommission Systems Costs

Figure 3.42: Decision Tree Alternatives

To help the decision maker visualize the decisions and options a decision tree was formed. Figure 3.42 describes at a high-level each of the alternatives the decision maker and stakeholders came up with and how their performance value and costs will be calculated.

Table 3.3: Status Quo Architecture Data Table



Availability Data		
Data Systems	Down Time (hours per year)	Total Time Elapsed (per year)
Finance	200	8760
HR	72	8760
PM	175	8760
Facilities	700	8760
Task Management	800	8760

Vulnerability Data										
Data Systems	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Finance	3	2	2	1	2	1	3	3	2	2
HR	2	2	1	2	3	2	1	2	3	3
PM	4	2	3	3	3	4	4	3	3	4
Facilities	2	2	2	4	2	2	3	4	2	2
Task Management	2	4	4	4	4	3	4	1	4	4

Query Data						
Data Systems	Q1	Q2	Q3	Q4	Q5	Q6
Finance	1	1	3	3	2	1
HR	1	1	4	4	4	1
PM	5	4	5	5	4	3
Facilities	1	2	3	2	3	1
Task Management	4	3	4	3	2	2

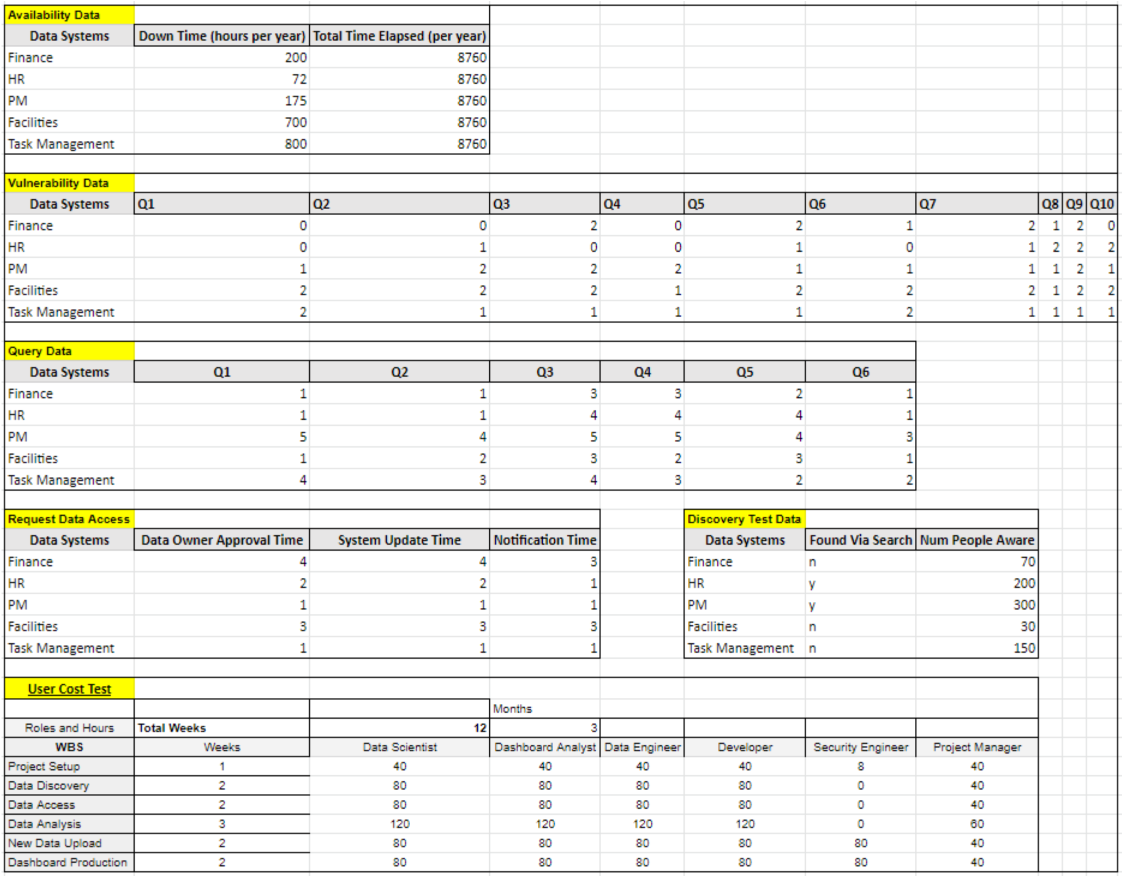
Request Data Access			
Data Systems	Data Owner Approval Time	System Update Time	Notification Time
Finance	4	4	3
HR	2	2	1
PM	1	1	1
Facilities	3	3	3
Task Management	1	1	1

Discovery Test Data		
Data Systems	Found Via Search	Num People Aware
Finance	n	70
HR	y	200
PM	y	300
Facilities	n	30
Task Management	n	150

User Cost Test							
			Months				
Roles and Hours	Total Weeks	12	3				
WBS	Weeks	Data Scientist	Dashboard Analyst	Data Engineer	Developer	Security Engineer	Project Manager
Project Setup	1	40	40	40	40	8	40
Data Discovery	2	80	80	80	80	0	40
Data Access	2	80	80	80	80	0	40
Data Analysis	3	120	120	120	120	0	60
New Data Upload	2	80	80	80	80	80	40
Dashboard Production	2	80	80	80	80	80	40

Alternative #1 Status Quo The current architecture is the status quo alternative. It represents what will happen if nothing is done. It is an evaluation of how the current architecture performs against the organization's objectives. It will be a baseline to measure how far off the organization currently is from their desired architecture. Table 3.3 is the data table containing all the information gathered from the different data systems in the current architecture. The data in this table will be used in the performance testing. The results of the performance testing for each of the data systems will then be aggregated according to the parametric diagrams.

Table 3.4: Status Quo Security Fix Architecture Data Table



Availability Data		
Data Systems	Down Time (hours per year)	Total Time Elapsed (per year)
Finance	200	8760
HR	72	8760
PM	175	8760
Facilities	700	8760
Task Management	800	8760

Vulnerability Data										
Data Systems	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Finance	0	0	2	0	2	1	2	1	2	0
HR	0	1	0	0	1	0	1	2	2	2
PM	1	2	2	2	1	1	1	1	2	1
Facilities	2	2	2	1	2	2	2	1	2	2
Task Management	2	1	1	1	1	2	1	1	1	1

Query Data						
Data Systems	Q1	Q2	Q3	Q4	Q5	Q6
Finance	1	1	3	3	2	1
HR	1	1	4	4	4	1
PM	5	4	5	5	4	3
Facilities	1	2	3	2	3	1
Task Management	4	3	4	3	2	2

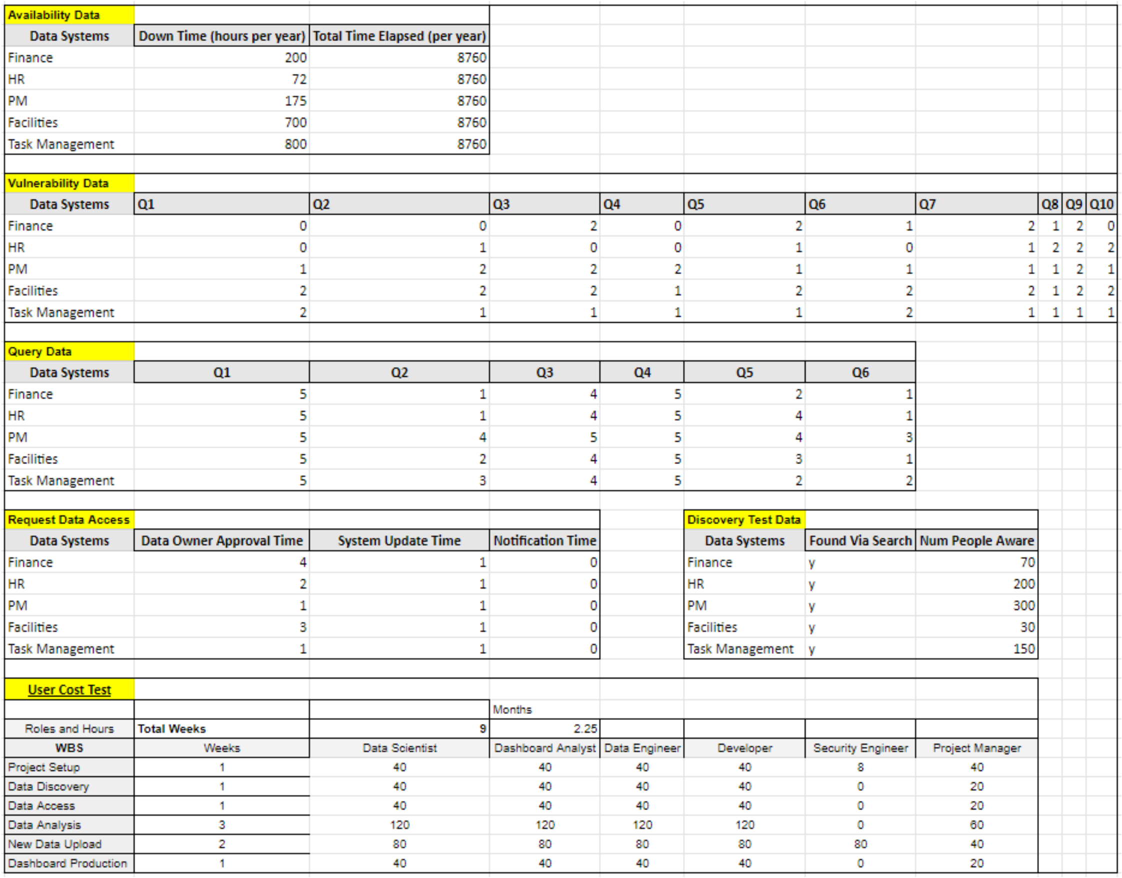
Request Data Access			
Data Systems	Data Owner Approval Time	System Update Time	Notification Time
Finance	4	4	3
HR	2	2	1
PM	1	1	1
Facilities	3	3	3
Task Management	1	1	1

Discovery Test Data		
Data Systems	Found Via Search	Num People Aware
Finance	n	70
HR	y	200
PM	y	300
Facilities	n	30
Task Management	n	150

User Cost Test							
			Months				
Roles and Hours	Total Weeks	12	3				
WBS	Weeks	Data Scientist	Dashboard Analyst	Data Engineer	Developer	Security Engineer	Project Manager
Project Setup	1	40	40	40	40	8	40
Data Discovery	2	80	80	80	80	0	40
Data Access	2	80	80	80	80	0	40
Data Analysis	3	120	120	120	120	0	60
New Data Upload	2	80	80	80	80	80	40
Dashboard Production	2	80	80	80	80	80	40

Alternative #2 Status Quo and Security Fix This alternative represents what would happen if all the organization decided to do was to fix the security vulnerabilities the EDA Initiative team discovered when gathering information on current architecture data systems. It is a decision to correct the issues on the current architecture but not invest in improving the data users ability to better access data. Table 3.4 is the data that reflects this option and will be used for the performance assessment. The difference between this data table and the status quo data table are the vulnerability questionnaire answers. It is assumed that the organization would want to fix all critical and major vulnerabilities immediately, so the new questionnaire reflects those fixes.

Table 3.5: Status Quo Data Updates Architecture Data Table



Availability Data

Data Systems	Down Time (hours per year)	Total Time Elapsed (per year)
Finance	200	8760
HR	72	8760
PM	175	8760
Facilities	700	8760
Task Management	800	8760

Vulnerability Data

Data Systems	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Finance	0	0	2	0	2	1	2	1	2	0
HR	0	1	0	0	1	0	1	2	2	2
PM	1	2	2	2	1	1	1	1	2	1
Facilities	2	2	2	1	2	2	2	1	2	2
Task Management	2	1	1	1	1	2	1	1	1	1

Query Data

Data Systems	Q1	Q2	Q3	Q4	Q5	Q6
Finance	5	1	4	5	2	1
HR	5	1	4	5	4	1
PM	5	4	5	5	4	3
Facilities	5	2	4	5	3	1
Task Management	5	3	4	5	2	2

Request Data Access

Data Systems	Data Owner Approval Time	System Update Time	Notification Time
Finance	4	1	0
HR	2	1	0
PM	1	1	0
Facilities	3	1	0
Task Management	1	1	0

Discovery Test Data

Data Systems	Found Via Search	Num People Aware
Finance	y	70
HR	y	200
PM	y	300
Facilities	y	30
Task Management	y	150

User Cost Test

			Months				
Roles and Hours	Total Weeks	9	2.25				
WBS	Weeks	Data Scientist	Dashboard Analyst	Data Engineer	Developer	Security Engineer	Project Manager
Project Setup	1	40	40	40	40	8	40
Data Discovery	1	40	40	40	40	0	20
Data Access	1	40	40	40	40	0	20
Data Analysis	3	120	120	120	120	0	60
New Data Upload	2	80	80	80	80	80	40
Dashboard Production	1	40	40	40	40	0	20

Alternative #3 Status Quo and Data Upgrades This alternative represents the option of improving the individual data systems instead of building a separate architecture. The security vulnerabilities would still need to be fixed, so the vulnerability data is the same as the Status Quo and Security Fix Alternative.

To improve the data systems performance the organization would direct all the data systems to do the following:

- Create an external API that data users could use to get data from the system.

- Provide a GUI Download option for data
- Would be required to get their data systems registered with the dashboard system and approved by security.
- Automate their process for providing a user access to the system, this should also send out notifications.
- Have their systems contact information and access process posted so it is discoverable by searching.

These improvements to the data systems will improve their performance scores. Table 3.5 are the data inputs for each of the data systems after they have made improvements listed above. The Data Discoverability, Request Data Access, and Query options scores will improve.

Alternative #4 EDA and Status Quo with Security Fixed This alternative represents the decision to build the future enterprise data architecture, but the current architecture will still exist in order to maintain the data systems functionality. This alternative is the combination of both the status quo and the future EDA system. It is assumed that the performance of this alternative will be the EDA value, because data users will no longer be going to the individual data systems for access, they will be going to the enterprise data architecture to get their data. The current architecture will undergo the same security fix procedure as the previous alternatives. Table 3.2 is the data table that will be used for the performance evaluation.

Alternative #5 EDA and Decommission Data Systems This alternative represents the decision to build the future enterprise data architecture and find data systems that can be decommissioned. Not all data systems will be able to be fully replaced by the enterprise data architecture and using other enterprise systems, but some data systems can be completely replaced. This is a way for the organization to consolidate after the EDA system is built and potentially reduce costs. The security issues in the current architecture will still need to be fixed, the EDA system will be built and in operation before data systems are evaluated and selected to be fully replaced and decommissioned by the new architecture. Table 3.2 is the data table that will be used for the performance evaluation. The performance of the architecture will be based on the EDA results since data users will be using EDA instead of the data systems directly. However the total cost of ownership will change because of the increase in cost for identifying and decommissioning a data system and the decreased cost for saving on the infrastructure and the data system's O&M costs.

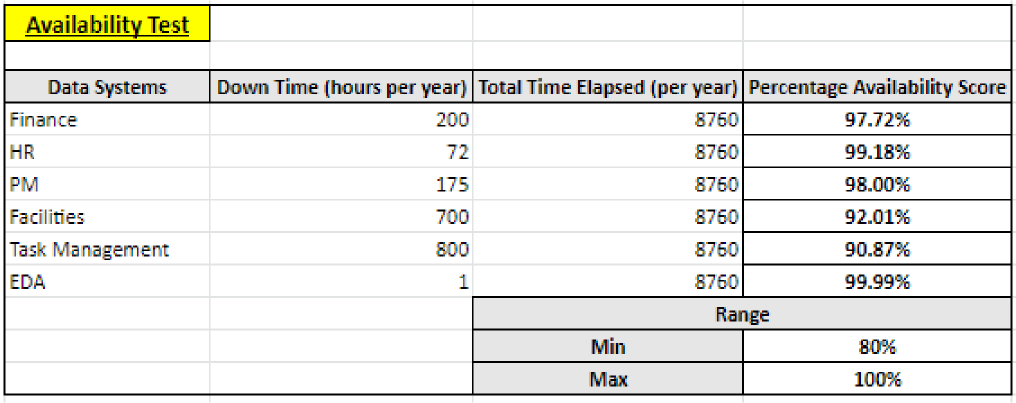
3.3.6 Architecture Performance Tests

The following tests were developed in order to evaluate the current and future data architectures on the performance of their designs. Because the future architecture doesn't exist yet, the tests were designed to fairly evaluate both the current architecture which does exist and the future one which is only a theoretical design whose input data can only be estimated. For the EDA results, they are estimated based on the EDA meeting all the requirements. Therefore scores for EDA are assumed to be more favorable.

These performance tests are meant to demonstrate how well each design meets the organizations objectives. Each alternative has input data collected for each one of the tests. The results of these test are the input data for the Single-Dimensional Value Functions (SDVFs). The ranges for the SDVFs will also come from the performance tests.

Refined use cases were selected for the performance tests. To measure how well each architecture makes data discoverable, the current architecture's activity diagram was used to describe the overall process. Figure 3.21 is the refined use case that will be used for the data discovery test because it is general enough to be applied to both architecture designs. Requesting data access to the system follows the same general workflow in both architecture designs, where once a data source has been identified and the data user knows how to request access, the process begins. The request data process will end when the data user receives a notification that they have access to the data.

Table 3.6: Performance Test for Availability



Availability Test			
Data Systems	Down Time (hours per year)	Total Time Elapsed (per year)	Percentage Availability Score
Finance	200	8760	97.72%
HR	72	8760	99.18%
PM	175	8760	98.00%
Facilities	700	8760	92.01%
Task Management	800	8760	90.87%
EDA	1	8760	99.99%
		Range	
		Min	80%
		Max	100%

Availability Test The Availability test is meant to measure the resiliency of the architecture. In order to do this it measures the architectures availability percentage over a year's time. To calculate this it requires how many hours the system was unavailable for and the number of hours it was supposed to be available for.

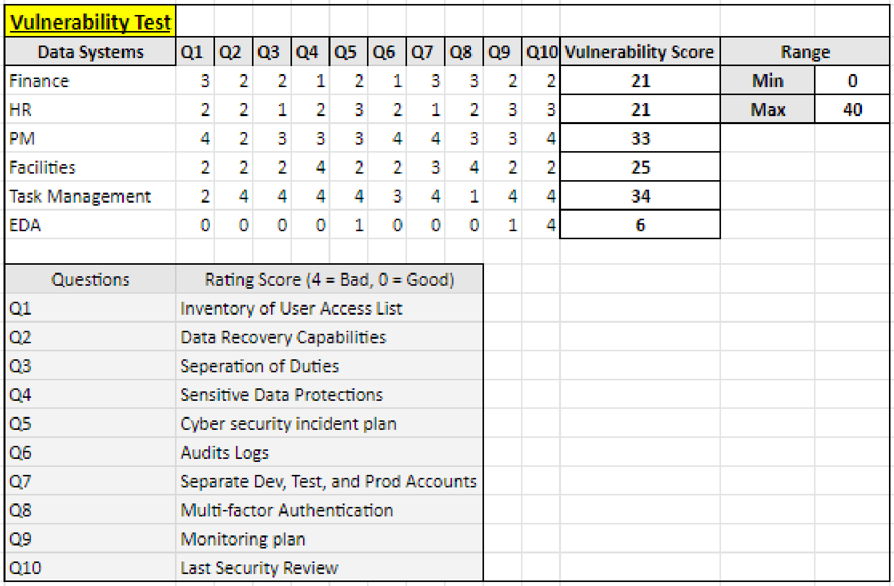
The following formula is used to calculate the availability percentage.

$$AvailabilityScore = \frac{ElapsedTime - DownTime}{ElapsedTime} * 100 \qquad (3.7)$$

Table 3.6 shows how the availability scores were calculated in excel. The range was determined by selecting the greatest possible value, 100% to be the upper-bound and 80% was selected to be the lower-bound.

Vulnerability Test The Vulnerability test was developed to measure how secure each architecture is. A series of questions were asked of each data system and the responses to each question were graded from a 0 to a 4. A score of 4 represents a critical vulnerability, a score of 3 represents a major vulnerability, a score of 2 represents a minor vulnerability, a score of 1 represents a low vulnerability, and a zero represents the best security score possible. The question scores are summed up to get a Vulnerability Score. Table 3.7 shows

Table 3.7: Performance Test for Vulnerability



Vulnerability Test

Data Systems	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Vulnerability Score	Range	
Finance	3	2	2	1	2	1	3	3	2	2	21	Min	0
HR	2	2	1	2	3	2	1	2	3	3	21	Max	40
PM	4	2	3	3	3	4	4	3	3	4	33		
Facilities	2	2	2	4	2	2	3	4	2	2	25		
Task Management	2	4	4	4	4	3	4	1	4	4	34		
EDA	0	0	0	0	1	0	0	0	1	4	6		

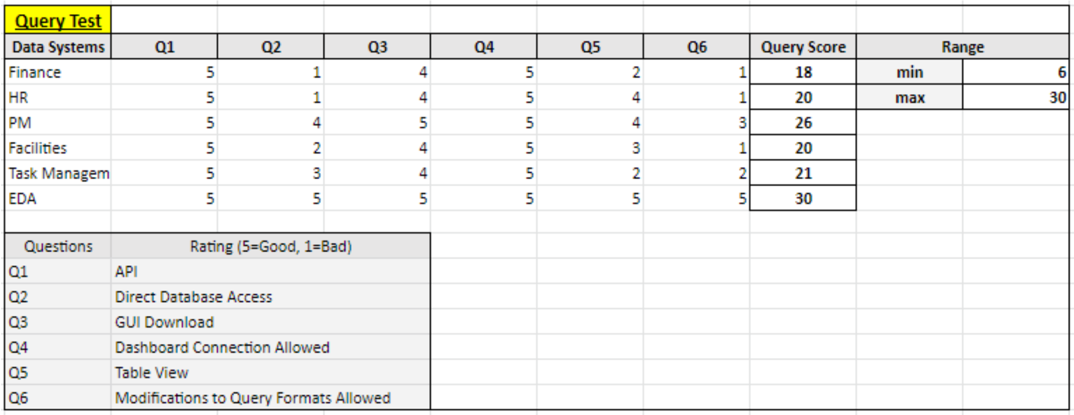
Questions	Rating Score (4 = Bad, 0 = Good)
Q1	Inventory of User Access List
Q2	Data Recovery Capabilities
Q3	Seperation of Duties
Q4	Sensitive Data Protections
Q5	Cyber security incident plan
Q6	Audits Logs
Q7	Separate Dev, Test, and Prod Accounts
Q8	Multi-factor Authentication
Q9	Monitoring plan
Q10	Last Security Review

how the vulnerability scores were calculated in excel. The range was determined by selecting the worst and best possible values to be the upper and lower bounds.

Query Options Test The Query Options test was developed to measure how well an architecture performed at providing methods for data users to acquire the data they need. A series of questions on query methods were asked of each data system, their responses were graded from 5 to 1. A score of 5 meant that the data system provided the most functionality possible with that method of query and a score of 1 meant they provided no functionality for that method of query. The Query Score is the sum of the query score responses. Table 3.8 shows how the query scores were calculated in excel. The range was determined by selecting the best possible option to be the upper-bound and 6 to be the lower-bound.

Data Access Request Test The Request Access Test was developed to measure how quickly an architecture performs at enabling data users to request access to the data they need. This test was modeled based on the refined use case for requesting access to data,

Table 3.8: Performance Test for Query Options

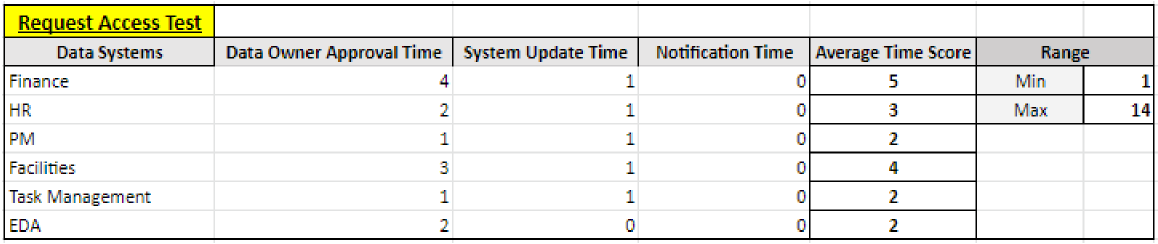


Query Test									
Data Systems	Q1	Q2	Q3	Q4	Q5	Q6	Query Score	Range	
Finance	5	1	4	5	2	1	18	min	6
HR	5	1	4	5	4	1	20	max	30
PM	5	4	5	5	4	3	26		
Facilities	5	2	4	5	3	1	20		
Task Managem	5	3	4	5	2	2	21		
EDA	5	5	5	5	5	5	30		

Questions	Rating (5=Good, 1=Bad)
Q1	API
Q2	Direct Database Access
Q3	GUI Download
Q4	Dashboard Connection Allowed
Q5	Table View
Q6	Modifications to Query Formats Allowed

which is broken into three parts similar to the current architecture activity diagram described in figure 3.22. The first part of the use case process is to contact the data owner and request permission. The second part is modifying the system so the user has access. The third part of the use case process is notifying the user they have access to the data. Each data team provided an estimated time frame on how long each part of the process takes on average. The Request Access Score is the sum of each part of the process.

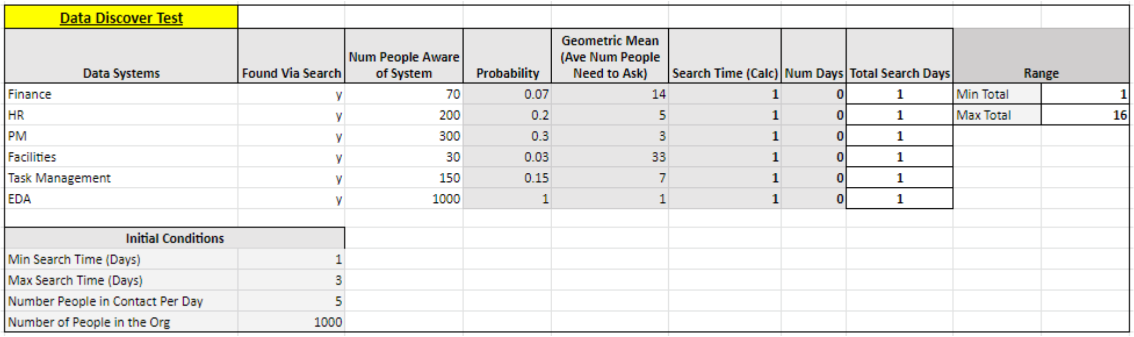
Table 3.9: Performance Test for Request Time



Request Access Test						
Data Systems	Data Owner Approval Time	System Update Time	Notification Time	Average Time Score	Range	
Finance	4	1	0	5	Min	1
HR	2	1	0	3	Max	14
PM	1	1	0	2		
Facilities	3	1	0	4		
Task Management	1	1	0	2		
EDA	2	0	0	2		

Table 3.9 shows how the query scores were calculated in excel. The range was determined by selecting the best possible score for the lower-bound and 14 for the upper-bound.

Table 3.10: Performance Test for Data Discovery



Data Discover Test									
Data Systems	Found Via Search	Num People Aware of System	Probability	Geometric Mean (Ave Num People Need to Ask)	Search Time (Calc)	Num Days	Total Search Days	Range	
Finance	y	70	0.07	14	1	0	1	Min Total	1
HR	y	200	0.2	5	1	0	1	Max Total	16
PM	y	300	0.3	3	1	0	1		
Facilities	y	30	0.03	33	1	0	1		
Task Management	y	150	0.15	7	1	0	1		
EDA	y	1000	1	1	1	0	1		
Initial Conditions									
Min Search Time (Days)	1								
Max Search Time (Days)	3								
Number People in Contact Per Day	5								
Number of People in the Org	1000								

Data Discovery Test The Data Discovery Test was developed to measure how quickly data users can discover the data they need and the data owner within the architecture so they can request access. This test was modeled based on the refined use case for discovering data, similar to the activity diagram in figure 3.21. A data user will first conduct a search to find the data and the data owner. If they don't discover it after a certain number of days they will start asking people they run into if they know who the data owner is and how to contact them. Each data system team was questioned and asked if they had posted information describing their data and how to request access to their data in a place that was discoverable by search within the architecture. And each data system was asked how many people in the organization would know the process for getting access to their data system. With both those pieces of information the Data Discovery Score is calculated in the following process.

If the data system does have information posted, then the Data Discovery score will be the minimum search time.

$$Searchable \left\{ DataDiscoveryScore = MinSearchTime \right.$$

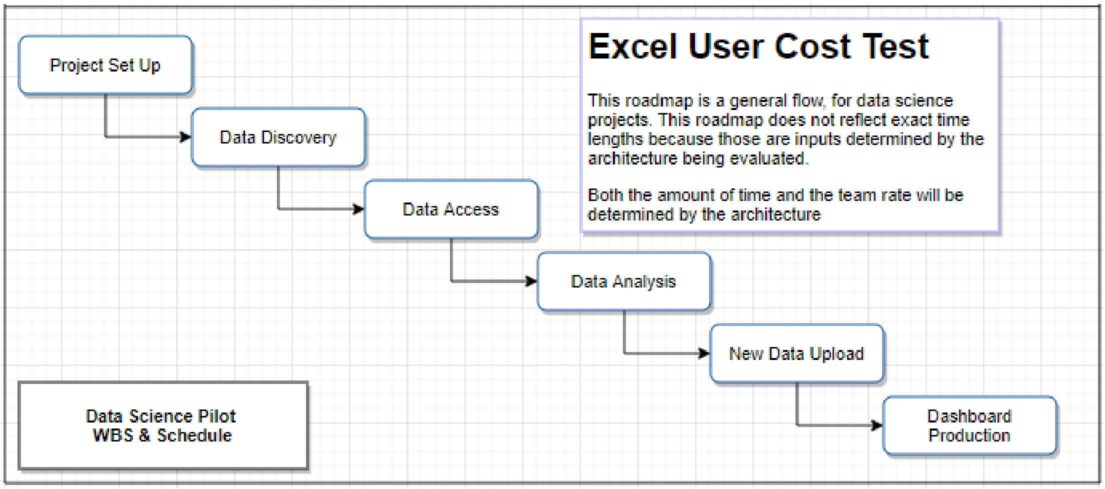
If they don't have their information posted their score is calculated by adding the max number of search days plus the number of days it takes to find a person who has the information they need. A geometric distribution mean is used to estimate the number of trials it will take to ask the right person assuming each person is independently selected. That geometric mean will be divided by the number of people who can be asked per day to get the Average Number of Days it takes to find someone who has the information a data user would need to request access. The probability used to calculate the geometric mean is based on the number of people in the organization who know how to get access to the data and the total number of people in the organization.

$$NotSearchable \begin{cases} P = \frac{NumPeopleAware}{TotalPeople} \\ AverageNumberofPeopleNeededtoAsk = GeometricDistributionMean = \frac{1}{P} \\ AverageNumberofDays = \frac{AverageNumberofPeopleNeededtoAsk}{NumberofPeopleAskedperDay} \\ DataDiscoveryScore = MaxSearchTime + AverageNumberofDays \end{cases}$$

This model is meant to represent the difficulty in finding data sources, when there isn't a data registry or catalog to search. This model makes assumptions that every person asked is an independent trail and the estimations provided by the data system teams on how many people know about their systems access process are accurate. The model isn't a perfect representation, but the decision maker accepted that this model as a good enough representation to grade the architectures with for this measure. If the decision maker still wasn't convinced it was a good enough representation, the performance test could be built to do a Monte Carlo simulation.

Table 3.10 shows how the Data Discovery scores were calculated in excel. The range was determined by selecting the best possible score for the lower-bound and 16 for the upper-bound.

User Cost Test The User Cost Test was developed to measure how reusable each architecture is and how much a data science project will cost under each architecture. This



Excel User Cost Test

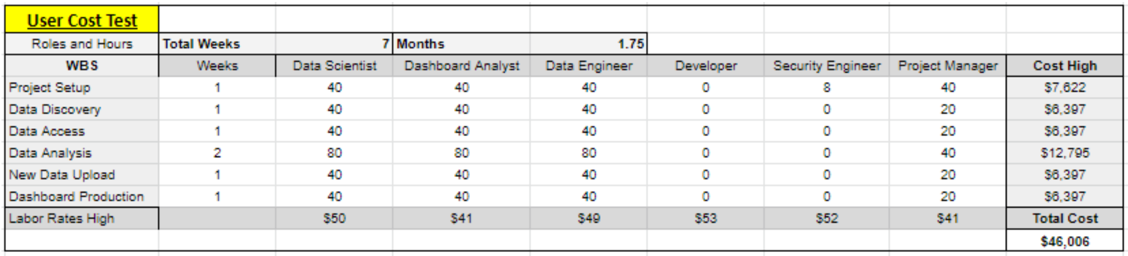
This roadmap is a general flow, for data science projects. This roadmap does not reflect exact time lengths because those are inputs determined by the architecture being evaluated.

Both the amount of time and the team rate will be determined by the architecture

Figure 3.43: Data Science Pilot Schedule

test was modeled based on the executive dashboard data science pilot. The cost will be determined based on how long it takes to complete each WBS task using the architecture. The labor rates will be the same, but the number of hours each team works and the staff working the project will change depending on the architecture. Figure 3.43 is the schedule of tasks the data science project will complete. To simplify the test, the project will assume a waterfall project management style.

Table 3.11: Performance Test for User Cost



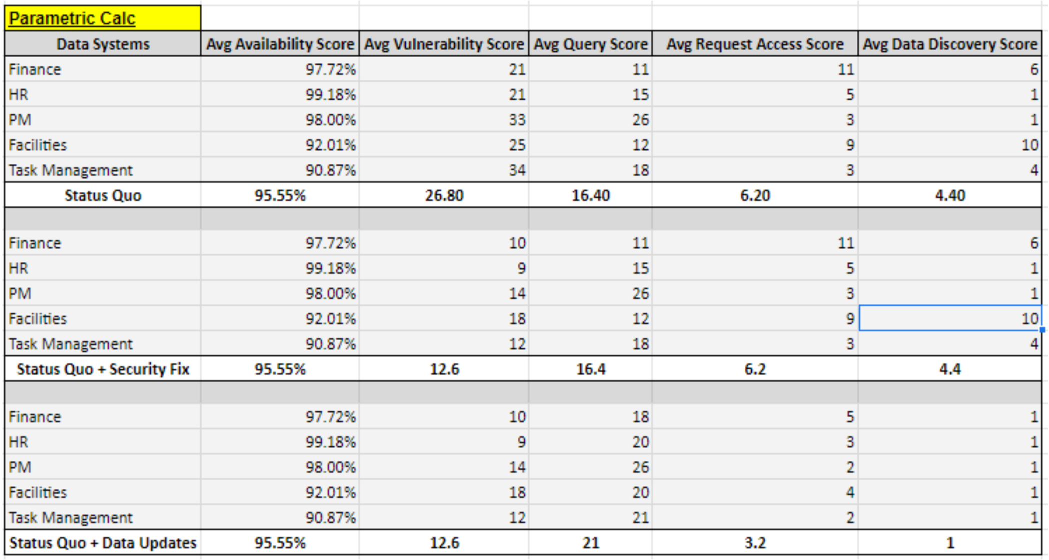
User Cost Test								
Roles and Hours	Total Weeks	7	Months	1.75				
WBS	Weeks	Data Scientist	Dashboard Analyst	Data Engineer	Developer	Security Engineer	Project Manager	Cost High
Project Setup	1	40	40	40	0	8	40	$7,622
Data Discovery	1	40	40	40	0	0	20	$6,397
Data Access	1	40	40	40	0	0	20	$6,397
Data Analysis	2	80	80	80	0	0	40	$12,795
New Data Upload	1	40	40	40	0	0	20	$6,397
Dashboard Production	1	40	40	40	0	0	20	$6,397
Labor Rates High		$50	$41	$49	$53	$52	$41	Total Cost
								$46,006

Table 3.11 shows how the User Cost scores were calculated in excel. The range was determined by selecting the best possible score for the lower-bound to be slightly less then

the alternatives best score. The upper-bound was set slightly higher than the worst possible score.

Parametric Current Architecture Calculations For the current architecture the performance scores need to be averaged in order to calculate the final scores. This averaging was represented graphically in the current architecture parametric diagram figure 3.23. For each of the current architecture alternatives the final scores will calculated in excel as shown in table 3.12.

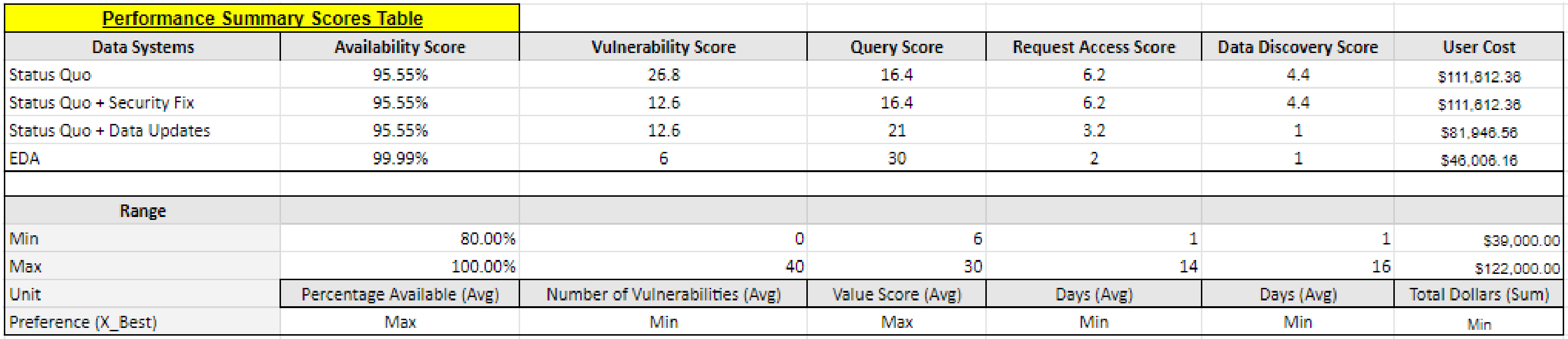
Table 3.12: Current Architecture Parametric Calculation



Parametric Calc					
Data Systems	Avg Availability Score	Avg Vulnerability Score	Avg Query Score	Avg Request Access Score	Avg Data Discovery Score
Finance	97.72%	21	11	11	6
HR	99.18%	21	15	5	1
PM	98.00%	33	26	3	1
Facilities	92.01%	25	12	9	10
Task Management	90.87%	34	18	3	4
Status Quo	95.55%	26.80	16.40	6.20	4.40
Finance	97.72%	10	11	11	6
HR	99.18%	9	15	5	1
PM	98.00%	14	26	3	1
Facilities	92.01%	18	12	9	10
Task Management	90.87%	12	18	3	4
Status Quo + Security Fix	95.55%	12.6	16.4	6.2	4.4
Finance	97.72%	10	18	5	1
HR	99.18%	9	20	3	1
PM	98.00%	14	26	2	1
Facilities	92.01%	18	20	4	1
Task Management	90.87%	12	21	2	1
Status Quo + Data Updates	95.55%	12.6	21	3.2	1

Performance Summary Scores All the performance scores for each of the alternatives are summarized in table 3.13. These are the scores that will be used in the single-dimensional value functions (SDVFs). The ranges, units, and preference direction are also listed for each of the SDVFs.

Table 3.13: Summary of the Performance Scores



Performance Summary Scores Table						
Data Systems	Availability Score	Vulnerability Score	Query Score	Request Access Score	Data Discovery Score	User Cost
Status Quo	95.55%	26.8	16.4	6.2	4.4	$111,612.36
Status Quo + Security Fix	95.55%	12.6	16.4	6.2	4.4	$111,612.36
Status Quo + Data Updates	95.55%	12.6	21	3.2	1	$81,946.56
EDA	99.99%	6	30	2	1	$46,006.16
Range						
Min	80.00%	0	6	1	1	$39,000.00
Max	100.00%	40	30	14	16	$122,000.00
Unit	Percentage Available (Avg)	Number of Vulnerabilities (Avg)	Value Score (Avg)	Days (Avg)	Days (Avg)	Total Dollars (Sum)
Preference (X_Best)	Max	Min	Max	Min	Min	Min

3.3.7 System Cost Model

Overview Cost Models are an estimate of the architecture's total cost. It will be based on the work breakdown structure (WBS) which takes into account the development work that needs to be done and the different stages of operation and maintenance on the architecture. There are different teams with different hourly rates and different team structures that will be building and maintaining the system. The cost will also include the infrastructure costs of the architecture. The total cost will be dependant on the length of time it takes to develop functions and each architecture alternative will be complete at different time lengths.

Calculating the costs of all the architecture alternatives is complicated and will require some assumptions to simplify the estimation. The result of the cost model should not be taken as the exact cost of what it will take to build the architecture, but it will be a good estimation for comparison between the alternatives. The resulting system cost will be used in the Single-Dimensional Value Function (SDVF) to normalize it for the final value-function calculation.

Data Collection To create the cost model for each of the alternatives the following data needs to be collected:

1. **Work Breakdown Structure (WBS)** For each alternative are the tasks to complete the development and maintain the operations of the architecture need to be included. The tasks are estimated at a high-level.

2. **Schedule** Based on the tasks that need to be accomplished the length of time to complete each of the tasks needs to be collected. Each of the schedules for the alternatives were designed with the waterfall method, to simplify the calculations and comparison. Each cost model will have a minimum schedule time and a maximum schedule time. The minimum schedule time will be based on that alternative's development time. The maximum schedule time will be an input variable the decision maker can change to see how costs change over a longer time period. The maximum schedule time will

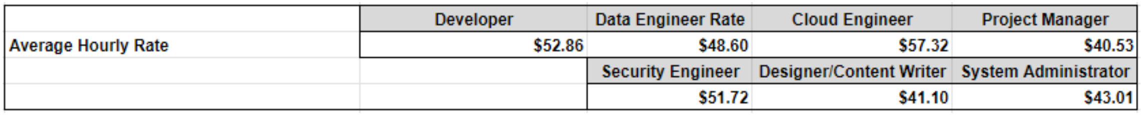
need to be at least as large or larger than the alternative with the longest development time. The maximum development time will be the same across all alternatives so they can be measured on the same timeline.

3. **Team Structure** Based on the work that needs to be done, teams need to be formed to accomplish that work. The teams should be staffed with the people that have the right skill set for the task. An alternative may have several teams working separate tasks of the WBS.

4. **Labor Rates** Each skill set comes with a different hourly rate which needs to be collected [66].

5. **Monthly Infrastructure Costs** For each architecture the infrastructure costs will be estimated on a monthly basis. The cost will include the development, test, and production accounts for the current and future architectures.

Table 3.14: Labor Rates



	Developer	Data Engineer Rate	Cloud Engineer	Project Manager
Average Hourly Rate	$52.86	$48.60	$57.32	$40.53
		Security Engineer	Designer/Content Writer	System Administrator
		$51.72	$41.10	$43.01

Table 3.14 shows the labor rates that were used to calculate all the different teams for each alternative. Each rate is the average hourly rate for each of the skill sets [66].

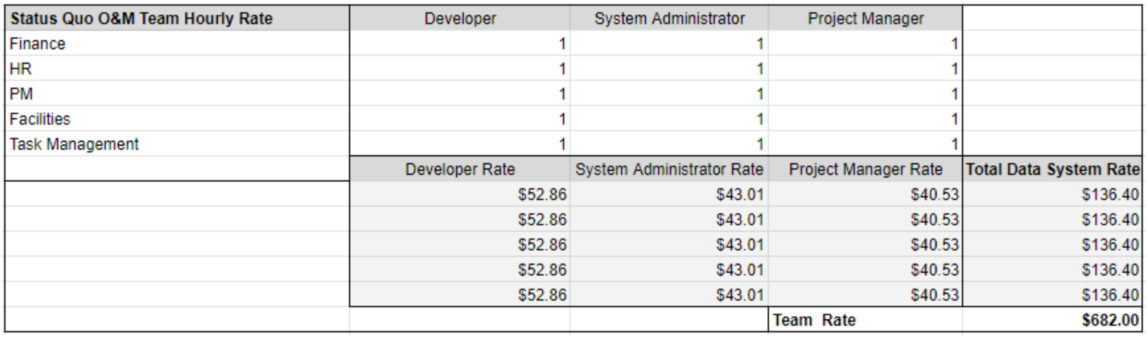
Current Architecture Data Collection Data was collected from each of the Data systems via interviews and documentation. Team structure and hours worked, infrastructure costs, and tasks were collected. The development work tasks and schedule for the alternatives are come from subject matter expertise estimations.

Future Architecture Data Collection Based on the research and design of the architecture subject matter experts estimated the data. The data is an estimation of the work tasks, skill sets needed, and time to complete the tasks. The infrastructure monthly costs were also estimated using subject matter experts.

Labor Costs Calculations The first calculation will be to estimate the labor costs for each of the alternatives. This will be done using the WBS, the schedule, and the team rates for each of the alternative architectures. The labor costs will then be combined with the infrastructure costs to get the total cost of the system at the end of this section.

Status Quo Labor Costs The Status Quo is the alternative that represents the current architecture which is comprised of several data systems that are maintained by separate teams. Each data system team has their own team structure. Table 3.15 shows the makeup of each data system team and sums up their hourly rates. The Status Quo team hourly rate will be use to estimate the O&M costs for the status quo, since there is no development work being done.

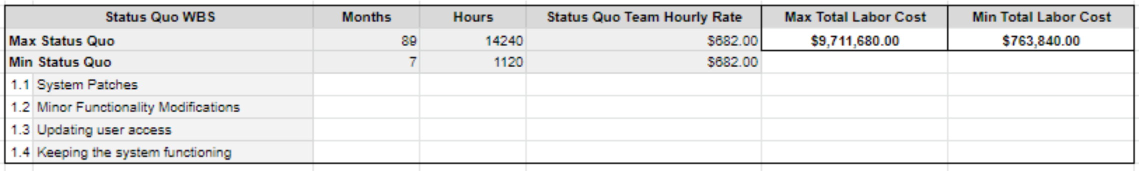
Table 3.15: Status Quo Team Hourly Rate



Status Quo O&M Team Hourly Rate	Developer	System Administrator	Project Manager	
Finance	1	1	1	
HR	1	1	1	
PM	1	1	1	
Facilities	1	1	1	
Task Management	1	1	1	
	Developer Rate	System Administrator Rate	Project Manager Rate	Total Data System Rate
	$52.86	$43.01	$40.53	$136.40
	$52.86	$43.01	$40.53	$136.40
	$52.86	$43.01	$40.53	$136.40
	$52.86	$43.01	$40.53	$136.40
	$52.86	$43.01	$40.53	$136.40
			Team Rate	$682.00

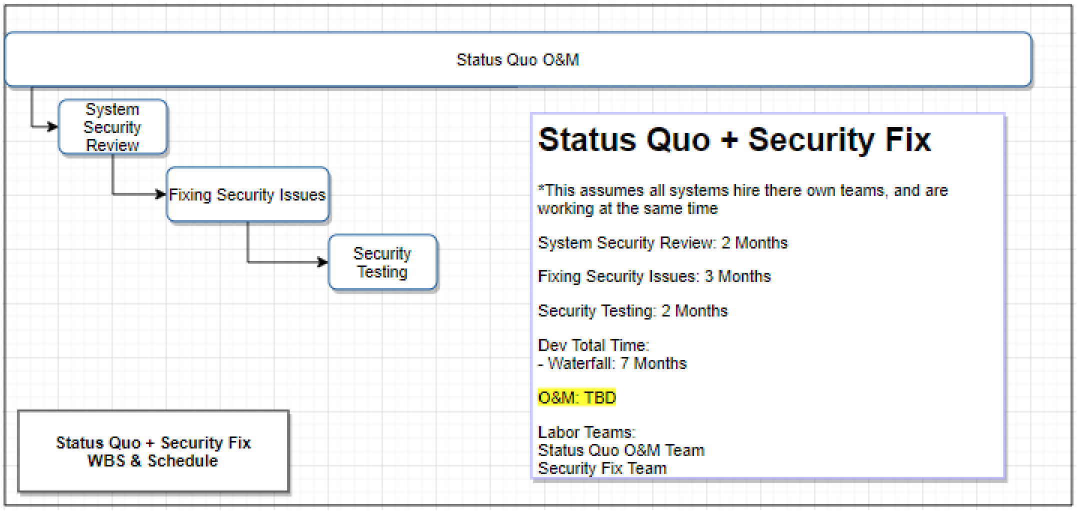
The schedule will be determined by the other alternatives, since there is no development work going on here. The minimum schedule length will be equal to the alternative with the minimum length and the maximum schedule length will be determined by the alternative with the maximum length. Table 3.16 shows how the labor costs for the status quo is calculated.

Table 3.16: Status Quo Labor Costs



Status Quo WBS	Months	Hours	Status Quo Team Hourly Rate	Max Total Labor Cost	Min Total Labor Cost
Max Status Quo	89	14240	$682.00	$9,711,680.00	$763,840.00
Min Status Quo	7	1120	$682.00		
1.1 System Patches					
1.2 Minor Functionality Modifications					
1.3 Updating user access					
1.4 Keeping the system functioning					

Status Quo + Security Fix Labor Costs The Status Quo and Security Fix alternative is similar to the status quo alternative except there is development work being done to fix the security issues. Figure 3.44 shows the WBS tasks and schedule to finish the development work. The full development work is estimated to take 7 months to finish, which will be the minimum schedule time for this alternative, when calculating the total cost. The maximum cost is calculated using a max time variable.



Status Quo + Security Fix

*This assumes all systems hire there own teams, and are working at the same time

System Security Review: 2 Months

Fixing Security Issues: 3 Months

Security Testing: 2 Months

Dev Total Time:
- Waterfall: 7 Months

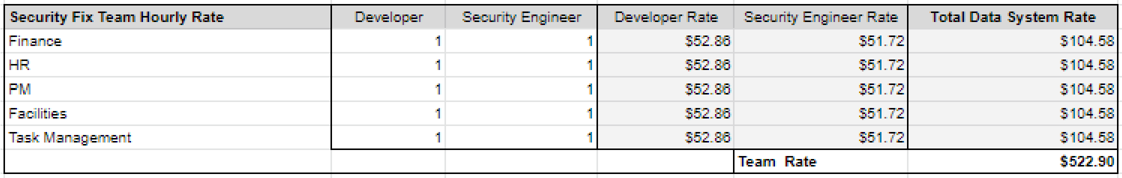
O&M: TBD

Labor Teams:
Status Quo O&M Team
Security Fix Team

Figure 3.44: Status Quo Security Fix Roadmap

There are two team hourly rates for the labor cost calculation. One team is the Status Quo O&M team and the other is the Security Fix team. Table 3.17 shows how the Security Fix team hourly rate is calculated.

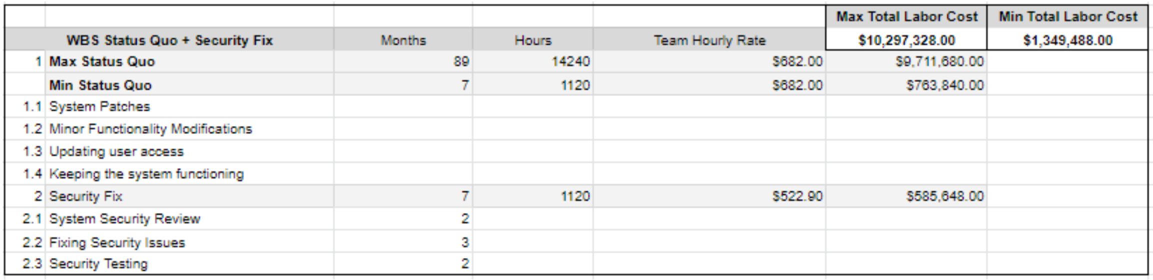
Table 3.17: Security Fix Team Rate



Security Fix Team Hourly Rate	Developer	Security Engineer	Developer Rate	Security Engineer Rate	Total Data System Rate
Finance	1	1	$52.86	$51.72	$104.58
HR	1	1	$52.86	$51.72	$104.58
PM	1	1	$52.86	$51.72	$104.58
Facilities	1	1	$52.86	$51.72	$104.58
Task Management	1	1	$52.86	$51.72	$104.58
				Team Rate	$522.90

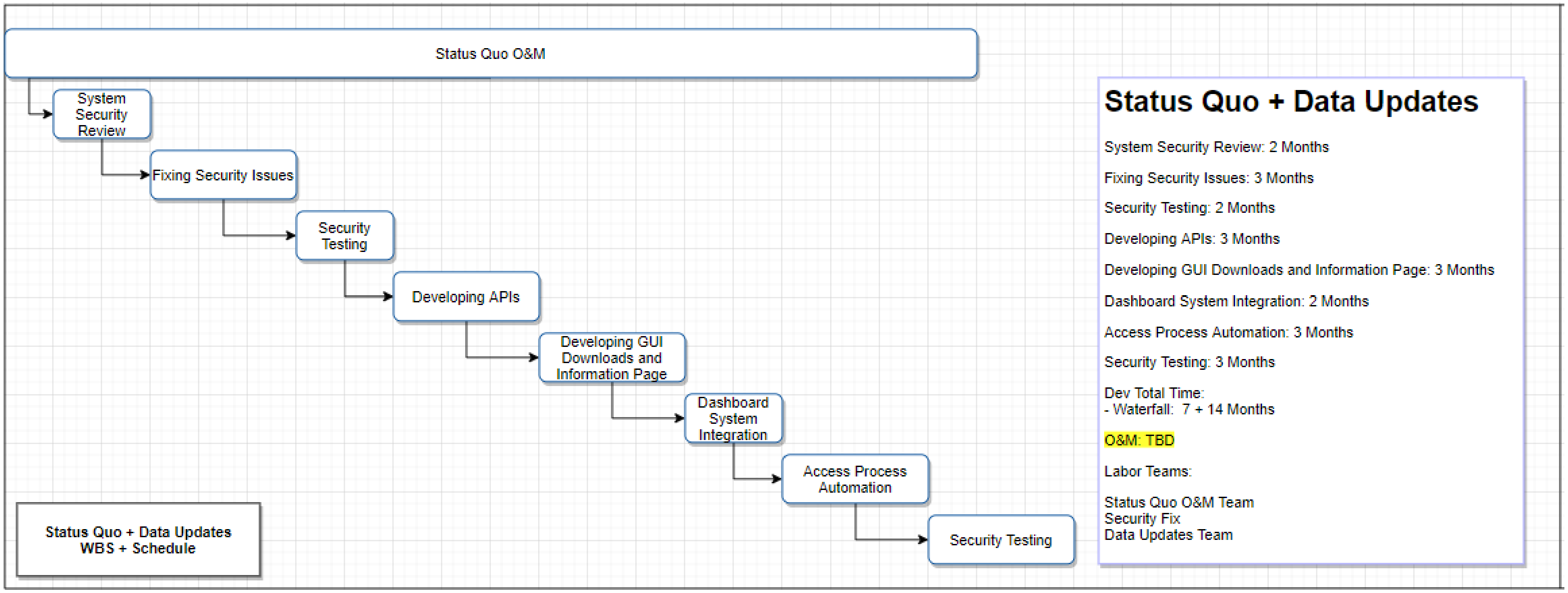
Table 3.18 shows how the labor costs are calculated for the Status Quo and Security Fix alternative in two parts. Part one calculates the cost of the O&M team and part two calculates the cost of the development work to fix the security issues by the Security Fix team. The two parts are summed to get the minimum and maximum costs.

Table 3.18: Status Quo Security Fix Labor Costs



					Max Total Labor Cost	Min Total Labor Cost
	WBS Status Quo + Security Fix	Months	Hours	Team Hourly Rate	$10,297,328.00	$1,349,488.00
1	Max Status Quo	89	14240	$682.00	$9,711,680.00	
	Min Status Quo	7	1120	$682.00	$763,840.00	
1.1	System Patches					
1.2	Minor Functionality Modifications					
1.3	Updating user access					
1.4	Keeping the system functioning					
2	Security Fix	7	1120	$522.90	$585,648.00	
2.1	System Security Review	2				
2.2	Fixing Security Issues	3				
2.3	Security Testing	2				

Status Quo + Data Upgrades Labor Costs The Status Quo and Data Updates alternative is an effort to make the current architecture better for data users without having to build a separate system. Each data system team will be tasked with making adjustments to their system so it supports data users outside of their current stakeholders and use cases. Each data system will hire a development team to fix the security issues and work the tasks outlined in section 3.3.5. Figure 3.45 shows the WBS tasks and schedule to finish the development work. The full development work is estimated to take 21 months to finish, which will be the minimum schedule time for this alternative, when calculating the total cost. The maximum cost is calculated using a max time variable.


Status Quo O&M
System Security Review
Fixing Security Issues
Security Testing
Developing APIs
Developing GUI Downloads and Information Page
Dashboard System Integration
Access Process Automation
Security Testing
Status Quo + Data Updates WBS + Schedule

Status Quo + Data Updates

System Security Review: 2 Months

Fixing Security Issues: 3 Months

Security Testing: 2 Months

Developing APIs: 3 Months

Developing GUI Downloads and Information Page: 3 Months

Dashboard System Integration: 2 Months

Access Process Automation: 3 Months

Security Testing: 3 Months

Dev Total Time:
- Waterfall: 7 + 14 Months

O&M: TBD

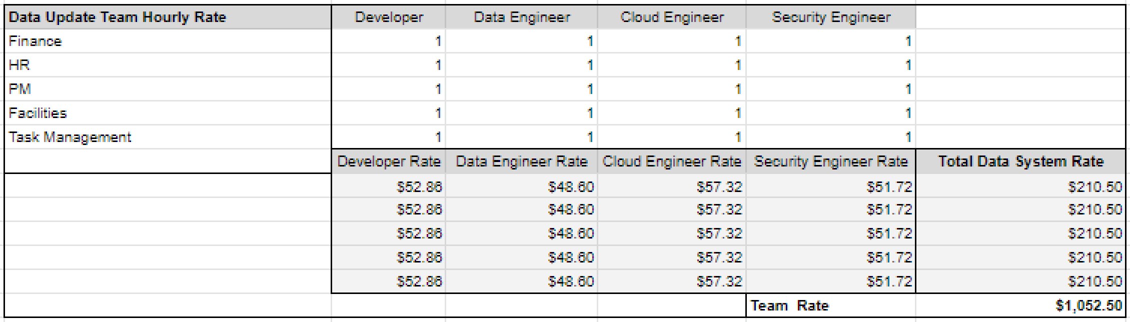
Labor Teams:

Status Quo O&M Team
Security Fix
Data Updates Team

Figure 3.45: Status Quo Data Updates Roadmaps

There are three team hourly rates for the labor cost calculation. One team is the Status Quo O&M team, the second is the Security Fix team, and third is the Data Update Development team. Table 3.19 shows how the Data Update team hourly rate is calculated.

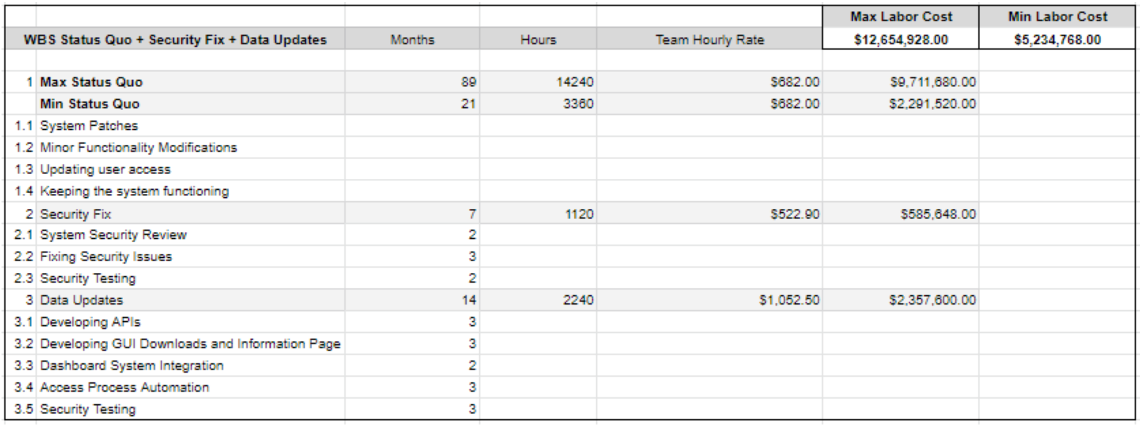
Table 3.19: Data Update Team Hourly Rate



Data Update Team Hourly Rate	Developer	Data Engineer	Cloud Engineer	Security Engineer	
Finance	1	1	1	1	
HR	1	1	1	1	
PM	1	1	1	1	
Facilities	1	1	1	1	
Task Management	1	1	1	1	
	Developer Rate	Data Engineer Rate	Cloud Engineer Rate	Security Engineer Rate	Total Data System Rate
	$52.86	$48.60	$57.32	$51.72	$210.50
	$52.86	$48.60	$57.32	$51.72	$210.50
	$52.86	$48.60	$57.32	$51.72	$210.50
	$52.86	$48.60	$57.32	$51.72	$210.50
	$52.86	$48.60	$57.32	$51.72	$210.50
				Team Rate	$1,052.50

Table 3.20 shows how the labor costs are calculated for the Status Quo and Data Updates alternative in three parts. Part one calculates the cost of the O&M team, part two calculates the cost of the development work to fix the security issues by the security fix team, and part three calculates the cost of development to upgrade the data systems by the data updates team. The three parts are summed to get the minimum and maximum costs.

Table 3.20: Status Quo Data Updates Labor Costs



					Max Labor Cost	Min Labor Cost
	WBS Status Quo + Security Fix + Data Updates	Months	Hours	Team Hourly Rate	$12,654,928.00	$5,234,768.00
1	Max Status Quo	89	14240	$682.00	$9,711,680.00	
	Min Status Quo	21	3360	$682.00	$2,291,520.00	
1.1	System Patches					
1.2	Minor Functionality Modifications					
1.3	Updating user access					
1.4	Keeping the system functioning					
2	Security Fix	7	1120	$522.90	$585,648.00	
2.1	System Security Review	2				
2.2	Fixing Security Issues	3				
2.3	Security Testing	2				
3	Data Updates	14	2240	$1,052.50	$2,357,600.00	
3.1	Developing APIs	3				
3.2	Developing GUI Downloads and Information Page	3				
3.3	Dashboard System Integration	2				
3.4	Access Process Automation	3				
3.5	Security Testing	3				

EDA + Status Quo + Security Fix Labor Costs The Enterprise Data Architecture and Status Quo plus Security Fix is the alternative to build the EDA and fix the current architecture's security issues. Figure 3.46 shows the WBS tasks and schedule to finish the development work. The security fix development and EDA development can overlap because the work is being done independently on separate systems. The full development work is estimated to take 18 months to finish, which will be the minimum schedule time for this alternative, when calculating the total cost. The maximum cost is calculated using a max time variable.

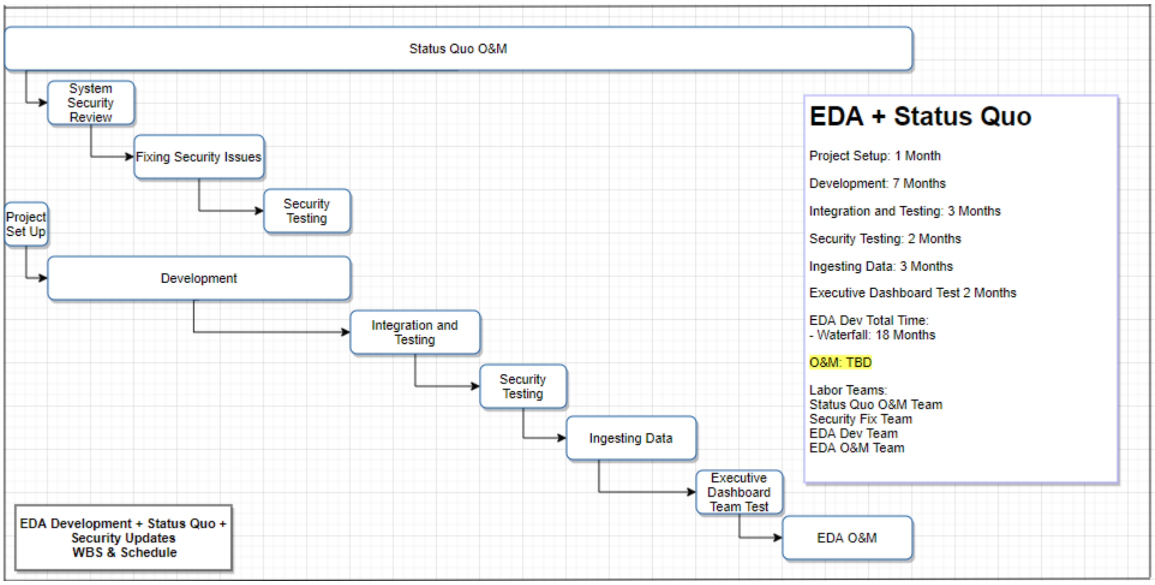
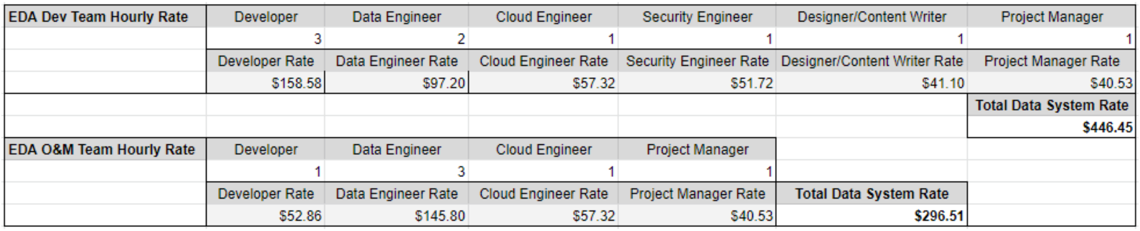


Figure 3.46: EDA Status Quo Roadmap

There are four team hourly rates for the labor cost calculation. One team is the Status Quo O&M team, the second is the Security Fix team, and third is the EDA Development team, and the fourth is the EDA O&M team. Table 3.21 shows how the EDA Development Team and EDA O&M team hourly rates were calculated.

Table 3.22 shows how the labor costs are calculated for the EDA and Status Quo plus Security Fix alternative in four parts. Part one calculates the cost of the Status Quo O&M

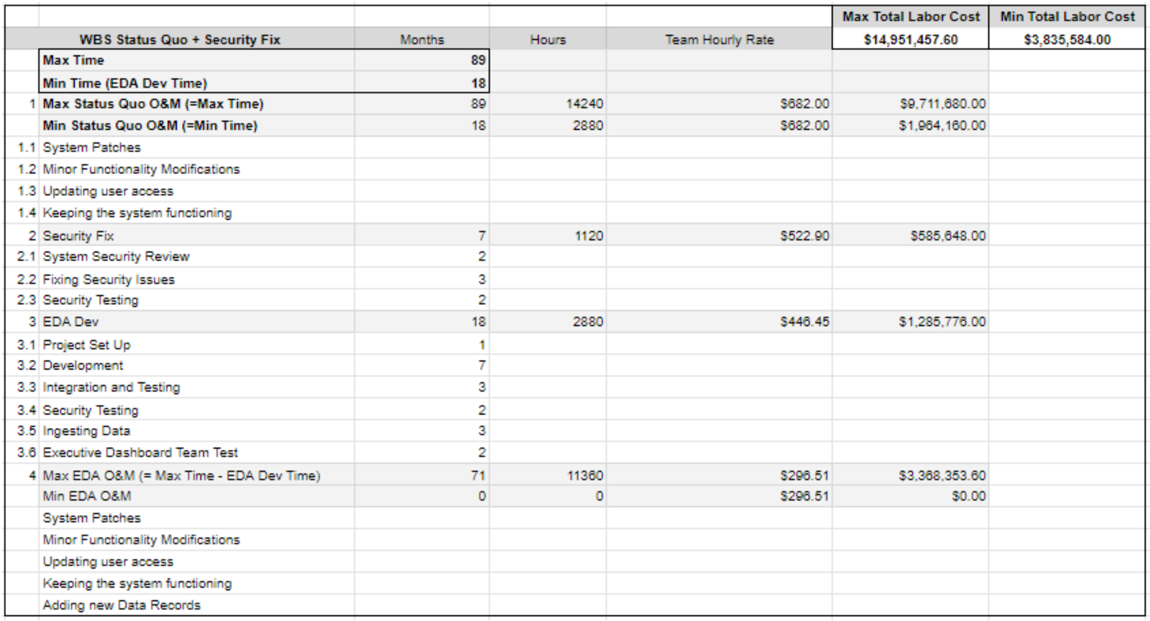
Table 3.21: EDA Dev Team Rate



EDA Dev Team Hourly Rate	Developer	Data Engineer	Cloud Engineer	Security Engineer	Designer/Content Writer	Project Manager
	3	2	1	1	1	1
	Developer Rate	Data Engineer Rate	Cloud Engineer Rate	Security Engineer Rate	Designer/Content Writer Rate	Project Manager Rate
	$158.58	$97.20	$57.32	$51.72	$41.10	$40.53
						Total Data System Rate
						$446.45
EDA O&M Team Hourly Rate	Developer	Data Engineer	Cloud Engineer	Project Manager		
	1	3	1	1		
	Developer Rate	Data Engineer Rate	Cloud Engineer Rate	Project Manager Rate	Total Data System Rate	
	$52.86	$145.80	$57.32	$40.53	$296.51	

team, part two calculates the cost of the development work to fix the security issues by the Security Fix team, and part three calculates the cost of development to build the EDA by the EDA Development team, and part four calculates the cost of the EDA O&M team. The four parts are summed to get the minimum and maximum costs. For the minimum cost, it does not include any EDA O&M team costs because it only accounts for the costs until the development is complete.

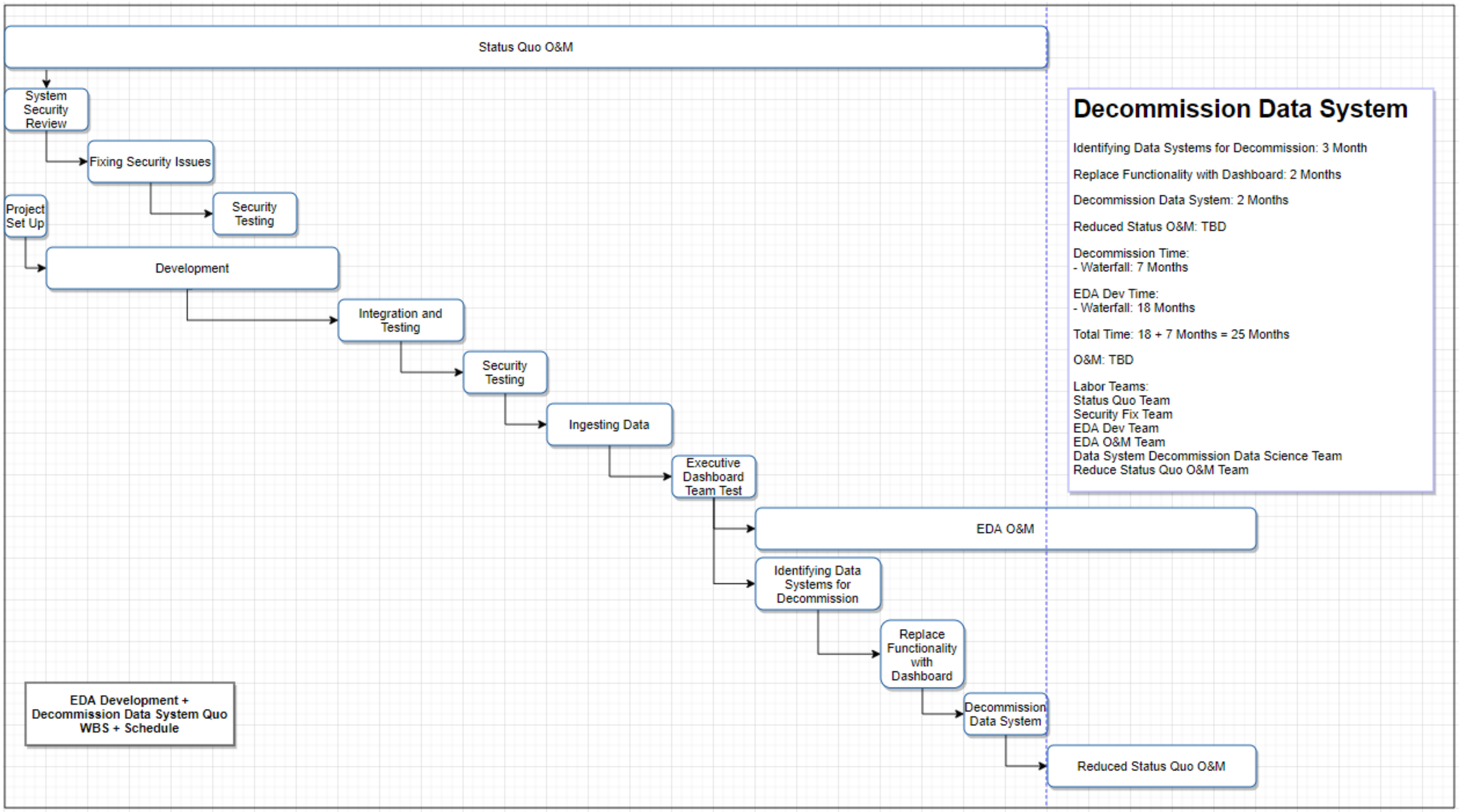
Table 3.22: EDA Status Quo Labor Costs



					Max Total Labor Cost	Min Total Labor Cost
	WBS Status Quo + Security Fix	Months	Hours	Team Hourly Rate	$14,951,457.60	$3,835,584.00
	Max Time	89				
	Min Time (EDA Dev Time)	18				
1	Max Status Quo O&M (=Max Time)	89	14240	$682.00	$9,711,680.00	
	Min Status Quo O&M (=Min Time)	18	2880	$682.00	$1,964,160.00	
1.1	System Patches					
1.2	Minor Functionality Modifications					
1.3	Updating user access					
1.4	Keeping the system functioning					
2	Security Fix	7	1120	$522.90	$585,648.00	
2.1	System Security Review	2				
2.2	Fixing Security Issues	3				
2.3	Security Testing	2				
3	EDA Dev	18	2880	$446.45	$1,285,776.00	
3.1	Project Set Up	1				
3.2	Development	7				
3.3	Integration and Testing	3				
3.4	Security Testing	2				
3.5	Ingesting Data	3				
3.6	Executive Dashboard Team Test	2				
4	Max EDA O&M (= Max Time - EDA Dev Time)	71	11360	$296.51	$3,368,353.60	
	Min EDA O&M	0	0	$296.51	$0.00	
	System Patches					
	Minor Functionality Modifications					
	Updating user access					
	Keeping the system functioning					
	Adding new Data Records					

EDA + Status Quo - Decommissioned Data System Labor Costs The Enterprise Data Architecture and Status Quo minus Decommission Data System is the alternative to build the EDA, fix the current architecture's security issues, and then identify data systems that can be fully replaced by the EDA and other enterprise systems. Figure 3.47 shows the WBS tasks and schedule to finish the development work. The full development work is estimated to take 25 months to finish, which will be the minimum schedule time for this alternative when calculating the total cost. The maximum cost is calculated using a max time variable.

There are six team hourly rates for the labor cost calculation. One team is the Status Quo O&M team, second is the Security Fix team, third is the EDA Development team, fourth is the EDA O&M team, fifth is the Decommission Team, and sixth is the Reduced Status Quo O&M team.



Decommission Data System

Identifying Data Systems for Decommission: 3 Month

Replace Functionality with Dashboard: 2 Months

Decommission Data System: 2 Months

Reduced Status O&M: TBD

Decommission Time:
- Waterfall: 7 Months

EDA Dev Time:
- Waterfall: 18 Months

Total Time: 18 + 7 Months = 25 Months

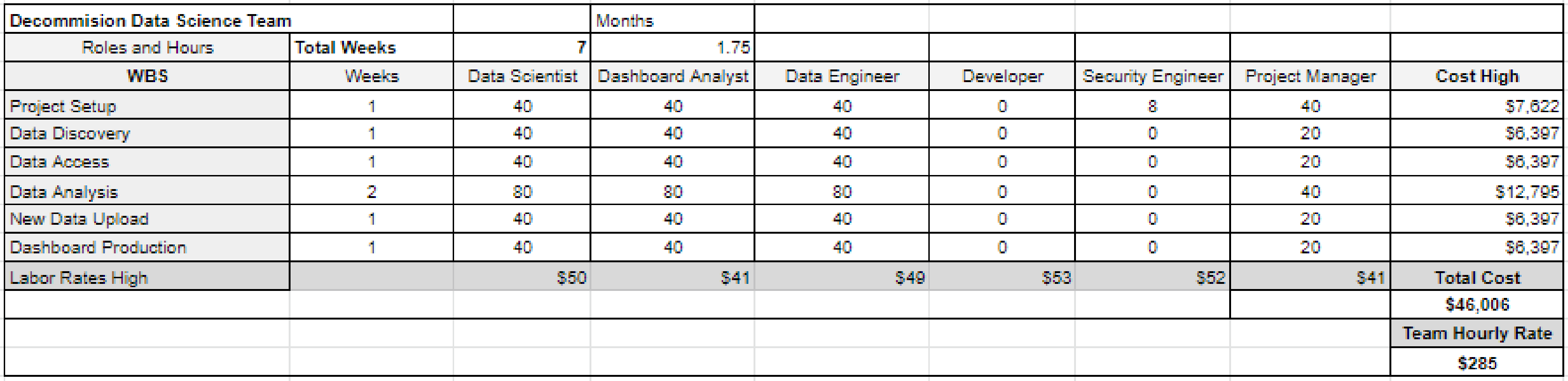
O&M: TBD

Labor Teams:
Status Quo Team
Security Fix Team
EDA Dev Team
EDA O&M Team
Data System Decommission Data Science Team
Reduce Status Quo O&M Team

Figure 3.47: EDA Decommission Status Quo Roadmap

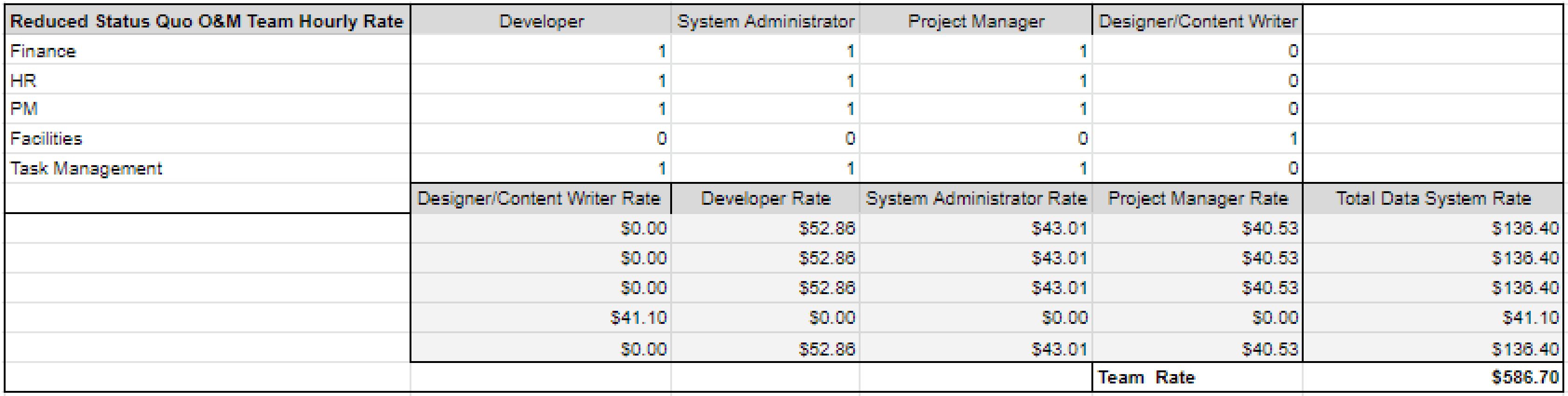
After the EDA system is built a data science team will be tasked with identifying a system that can be replaced with EDA and other enterprise systems. Table 3.23 shows how the Decommission Data Science team hourly rate is calculated. It is the same skill set and hourly rates as the User Cost calculation. It assumes one person of each skill set will be on the team.

Table 3.23: EDA Decommission Team Hourly Rate



Decommision Data Science Team			Months					
Roles and Hours	**Total Weeks**	7	1.75					
WBS	Weeks	Data Scientist	Dashboard Analyst	Data Engineer	Developer	Security Engineer	Project Manager	**Cost High**
Project Setup	1	40	40	40	0	8	40	$7,622
Data Discovery	1	40	40	40	0	0	20	$6,397
Data Access	1	40	40	40	0	0	20	$6,397
Data Analysis	2	80	80	80	0	0	40	$12,795
New Data Upload	1	40	40	40	0	0	20	$6,397
Dashboard Production	1	40	40	40	0	0	20	$6,397
Labor Rates High		$50	$41	$49	$53	$52	$41	**Total Cost**
								$46,006
								Team Hourly Rate
								$285

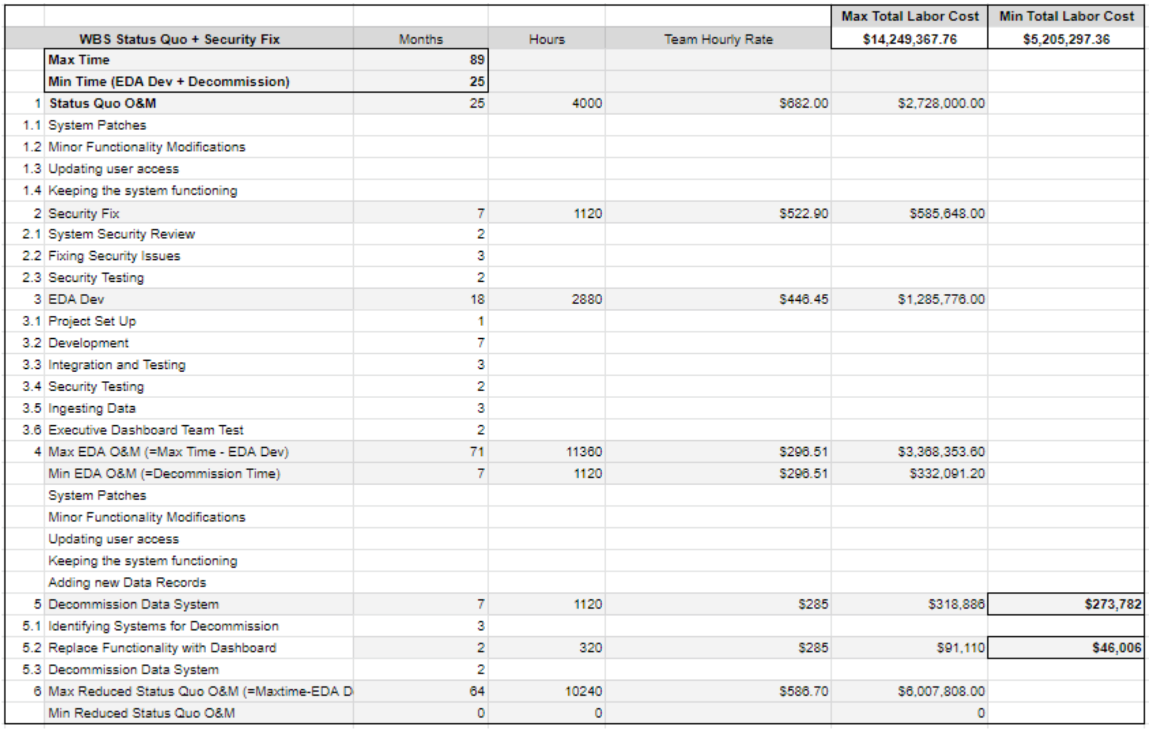
Table 3.24: EDA Reduce Status Quo Hourly Rate



Reduced Status Quo O&M Team Hourly Rate	Developer	System Administrator	Project Manager	Designer/Content Writer	
Finance	1	1	1	0	
HR	1	1	1	0	
PM	1	1	1	0	
Facilities	0	0	0	1	
Task Management	1	1	1	0	
	Designer/Content Writer Rate	Developer Rate	System Administrator Rate	Project Manager Rate	Total Data System Rate
	$0.00	$52.86	$43.01	$40.53	$136.40
	$0.00	$52.86	$43.01	$40.53	$136.40
	$0.00	$52.86	$43.01	$40.53	$136.40
	$41.10	$0.00	$0.00	$0.00	$41.10
	$0.00	$52.86	$43.01	$40.53	$136.40
				Team Rate	**$586.70**

The Decommission Data Science team will select the facilities data system to be decommissioned. They determined the systems functionality can be completely replaced with the dashboarding system and EDA. Once the facility dashboards have been built and tested, the system will be decommissioned and the Status Quo O&M rate will be adjusted so it no longer includes the facilities labor cost. Table 3.24 shows how the Reduced Status Quo O&M team hourly rate is calculated.

Table 3.25: EDA Decommission Labor Costs



					Max Total Labor Cost	Min Total Labor Cost
	WBS Status Quo + Security Fix	Months	Hours	Team Hourly Rate	$14,249,367.76	$5,205,297.36
	Max Time	89				
	Min Time (EDA Dev + Decommission)	25				
1	Status Quo O&M	25	4000	$682.00	$2,728,000.00	
1.1	System Patches					
1.2	Minor Functionality Modifications					
1.3	Updating user access					
1.4	Keeping the system functioning					
2	Security Fix	7	1120	$522.90	$585,648.00	
2.1	System Security Review	2				
2.2	Fixing Security Issues	3				
2.3	Security Testing	2				
3	EDA Dev	18	2880	$446.45	$1,285,776.00	
3.1	Project Set Up	1				
3.2	Development	7				
3.3	Integration and Testing	3				
3.4	Security Testing	2				
3.5	Ingesting Data	3				
3.6	Executive Dashboard Team Test	2				
4	Max EDA O&M (=Max Time - EDA Dev)	71	11360	$296.51	$3,368,353.60	
	Min EDA O&M (=Decommission Time)	7	1120	$296.51	$332,091.20	
	System Patches					
	Minor Functionality Modifications					
	Updating user access					
	Keeping the system functioning					
	Adding new Data Records					
5	Decommission Data System	7	1120	$285	$318,886	$273,782
5.1	Identifying Systems for Decommission	3				
5.2	Replace Functionality with Dashboard	2	320	$285	$91,110	$46,006
5.3	Decommission Data System	2				
6	Max Reduced Status Quo O&M (=Maxtime-EDA D	64	10240	$586.70	$6,007,808.00	
	Min Reduced Status Quo O&M	0	0		0	

Table 3.25 shows how the labor costs are calculated for the EDA and Status Quo minus Decommission Data System alternative in seven parts. Part one calculates the cost of the Status Quo O&M team. Part two calculates the cost of the development work to fix the security issues by the Security Fix team. Part three calculates the cost of development to build the EDA by the EDA Development team. Part four and five calculates the cost of

the decommission work, which includes the development of the dashboards, by the Decommission Data Science team. Because the data science team is using the EDA architecture to build the dashboards, the cost is the same User Cost results for the EDA, which is why it's broken out in the calculation. Part six calculates the cost of the EDA O&M team. Part seven calculates the cost of the Reduces Status Quo O&M team. The seven parts are summed to get the minimum and maximum costs. For the minimum cost, it does not include any Reduced Status Quo O&M team costs because it only accounts for the costs until the development is complete.

Infrastructure Costs Both the current architecture and EDA are built with a cloud service provider that charges on a monthly basis. This makes it easier to calculate costs for all development, test, and production accounts of both architectures. However there are still several assumptions being made to simplify calculations and make the comparison equitable. One assumption being made is that the dev, test, and prod accounts will require the same architecture to be deployed 24/7, but the dev and test accounts won't need instances as large as the prod accounts. Since the cloud providers cost model is pay for use, the calculations do not account for additional services that might be used during the development process, only the architecture itself. It also assumes that the infrastructure cost is consistent throughout the development process and does not take into account any of the free tier offerings.

The Appendix A shows the data tables from which all the architectures will utilize for their calculations. The data was taken from AWS website [70]. There are assumptions made that a certain instance type is perfectly adequate for the type of work and demand. Since the same data table is being used across all the data systems it should minimize the bias in estimating the cost for the infrastructure. These costs are not meant to be true estimations, but good enough for comparison purposes.

Status Quo Infrastructure Costs The current architecture is made up of several individual data systems that maintain their own accounts. Each data system has their own architecture with development, test, and production accounts. Table A.1 in the appendix shows each of the data systems costs for all components of their architecture in all their accounts. The total monthly infrastructure cost is the sum of all the data systems monthly total costs.

Status Quo + Security Fix Infrastructure Costs To account for the security fix modifications made to the data systems, the dev and test costs were double. This assumption was made because the data systems should be doing more development and testing of the systems, but the infrastructure components would not change. Table A.2 in the appendix shows each of the data systems costs for all components of their architecture in all there accounts. The total monthly infrastructure cost is the sum of all the data systems monthly total costs.

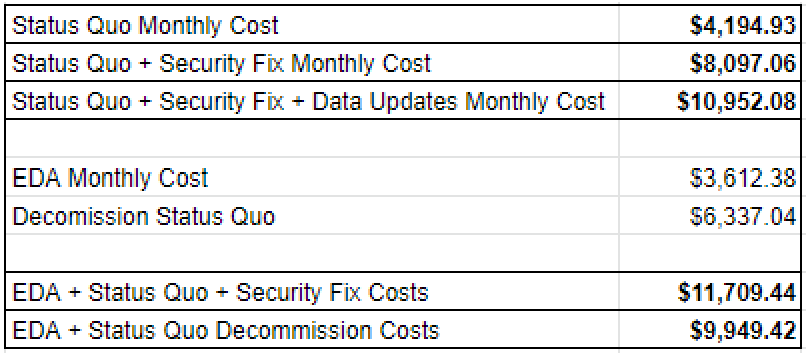
Status Quo + Data Upgrades Infrastructure Costs Modifications will need to be made to the data systems to support the data upgrade requirements outlined in section 3.3.5. Data systems will have a new server added that handles the external API query requests. To support the automation of adding users to the system and making the data discoverable, two more servers were added. Each system built a website that provided information about the system and a back-end server automates the process for adding and notifying the data user. In total three servers were added with additional EBS storage local storage to all accounts. Table A.2 in the appendix shows each of the data systems costs for all components of their architecture in all there accounts. The total monthly infrastructure cost is the sum of all the data systems monthly total costs.

Status Quo - Decommissioned Data System Infrastructure Costs The infrastructure cost for the Status Quo minus the Decommissioned Data System is the same as the security fix cost, but without the cost for the facilities data system. Table A.4in the appendix shows each of the data systems infrastructure costs, except for the facilities data system, and calculates the total monthly infrastructure cost, which is the sum of the data systems costs.

EDA Infrastructure Costs The EDA infrastructure cost is one system that has a development, test, and production account. Components of the architecture are grouped by the EDA subsystems, but all the components are in the same dev, test, and prod accounts. Table A.5 in the appendix shows each of the component costs. The EDA total cost is the sum of all the component costs. This is the same calculation that is depicted in the parametric diagram figure 3.41.

Infrastructure Summary Costs Two of the alternatives will have combined infrastructure monthly costs. Table 3.26 shows how the EDA alternative infrastructure costs are combined for a final infrastructure monthly cost. These costs are what will be combined with the labor costs to calculate the total cost of ownership for each architecture.

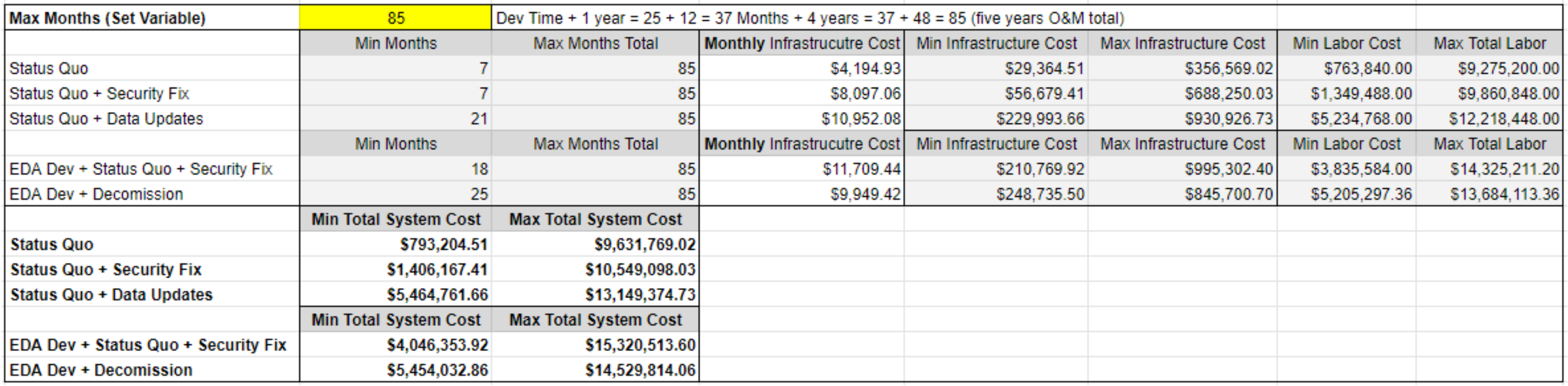
Table 3.26: Infrastructure Costs Summery



Status Quo Monthly Cost	**$4,194.93**
Status Quo + Security Fix Monthly Cost	**$8,097.06**
Status Quo + Security Fix + Data Updates Monthly Cost	**$10,952.08**
EDA Monthly Cost	$3,612.38
Decomission Status Quo	$6,337.04
EDA + Status Quo + Security Fix Costs	**$11,709.44**
EDA + Status Quo Decommission Costs	**$9,949.42**

Combined Cost Model The data from the labor costs and infrastructure costs for the alternatives are brought together to calculate the total cost of ownership. Table 3.27 shows the table of how everything is combined. Each alternative has a minimum cost that is calculated using it's minimum schedule, which is based on the time it takes to complete it's development. Each alternative's maximum cost is based on the same maximum time value, this is so each alternative can be estimated on the same timeline. The infrastructure monthly cost is multiplied by the min and max time to calculate the min and max infrastructure costs. The min and max infrastructure costs are added to the min and max labor costs to calculate the final min and max total costs for each alternative.

Table 3.27: Total Cost Data Summary Table



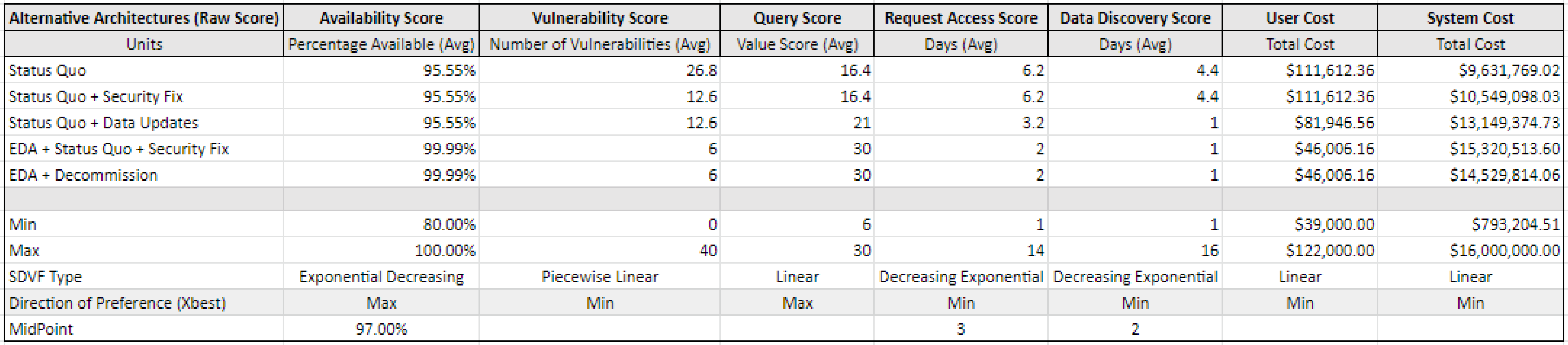
Max Months (Set Variable)	85	Dev Time + 1 year = 25 + 12 = 37 Months + 4 years = 37 + 48 = 85 (five years O&M total)						
	Min Months	Max Months Total	**Monthly** Infrastrucutre Cost	Min Infrastructure Cost	Max Infrastructure Cost	Min Labor Cost	Max Total Labor
Status Quo	7	85	$4,194.93	$29,364.51	$356,569.02	$763,840.00	$9,275,200.00
Status Quo + Security Fix	7	85	$8,097.06	$56,679.41	$688,250.03	$1,349,488.00	$9,860,848.00
Status Quo + Data Updates	21	85	$10,952.08	$229,993.66	$930,926.73	$5,234,768.00	$12,218,448.00
	Min Months	Max Months Total	**Monthly** Infrastrucutre Cost	Min Infrastructure Cost	Max Infrastructure Cost	Min Labor Cost	Max Total Labor
EDA Dev + Status Quo + Security Fix	18	85	$11,709.44	$210,769.92	$995,302.40	$3,835,584.00	$14,325,211.20
EDA Dev + Decomission	25	85	$9,949.42	$248,735.50	$845,700.70	$5,205,297.36	$13,684,113.36
	Min Total System Cost	**Max Total System Cost**					
Status Quo	**$793,204.51**	**$9,631,769.02**					
Status Quo + Security Fix	**$1,406,167.41**	**$10,549,098.03**					
Status Quo + Data Updates	**$5,464,761.66**	**$13,149,374.73**					
	Min Total System Cost	**Max Total System Cost**					
EDA Dev + Status Quo + Security Fix	**$4,046,353.92**	**$15,320,513.60**					
EDA Dev + Decomission	**$5,454,032.86**	**$14,529,814.06**					

3.3.8 Experiment Framework Model

This final section will be pulling together the alternative's data from all the sub-models into a consequence table. The data will then be normalized using the single-dimensional value functions. The results will then be used in the value function along with the weights for the objectives to calculate the final value for the alternatives. These values can then show how the alternative architecture implementations rank against each other based on the data from the sub-models and the weights of the objectives.

Alternative's Data Table 3.28 is a table combining all the data from the performance tests and cost models for each of the alternatives. It shows how each alternative compares to one another before they are normalized. The ranges, preference direction, midpoints, and SDVF type are also shown for reference. This table is useful for summarizing the results of the sub-model work.

Table 3.28: Alternative Data Score Summary



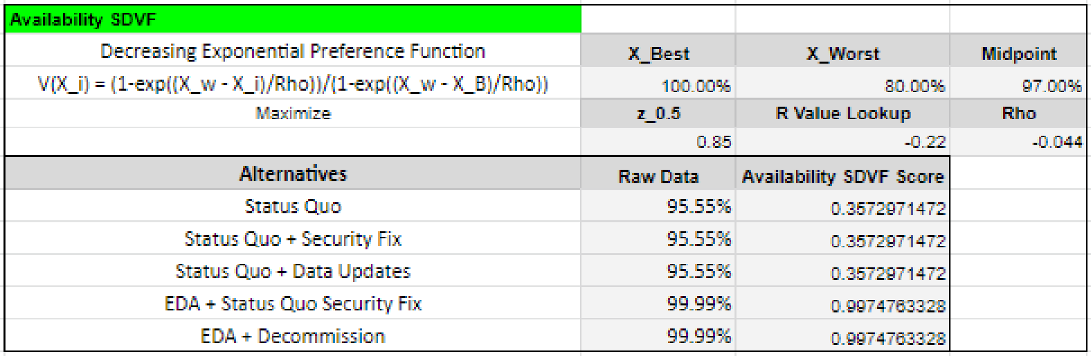
Alternative Architectures (Raw Score)	Availability Score	Vulnerability Score	Query Score	Request Access Score	Data Discovery Score	User Cost	System Cost
Units	Percentage Available (Avg)	Number of Vulnerabilities (Avg)	Value Score (Avg)	Days (Avg)	Days (Avg)	Total Cost	Total Cost
Status Quo	95.55%	26.8	16.4	6.2	4.4	$111,612.36	$9,631,769.02
Status Quo + Security Fix	95.55%	12.6	16.4	6.2	4.4	$111,612.36	$10,549,098.03
Status Quo + Data Updates	95.55%	12.6	21	3.2	1	$81,946.56	$13,149,374.73
EDA + Status Quo + Security Fix	99.99%	6	30	2	1	$46,006.16	$15,320,513.60
EDA + Decommission	99.99%	6	30	2	1	$46,006.16	$14,529,814.06
Min	80.00%	0	6	1	1	$39,000.00	$793,204.51
Max	100.00%	40	30	14	16	$122,000.00	$16,000,000.00
SDVF Type	Exponential Decreasing	Piecewise Linear	Linear	Decreasing Exponential	Decreasing Exponential	Linear	Linear
Direction of Preference (Xbest)	Max	Min	Max	Min	Min	Min	Min
MidPoint	97.00%			3	2		

Single Dimensional Value Function Calculations The SDVF will take in input from each of the alternatives and output a normalized value based on the decision maker's preferences which are built into the equations. Each equation type represents how a decision maker's preference changes over the range of values. The best possible outcome of an SDVF will be a 1, which represents the "most value" something could have for the decision maker. The worst possible outcome of an SDVF will be a 0, which represents "no value" for the decision maker.

Rho Calculation For the exponential preference functions, the Rho value needs to be calculated first. Using the midpoint, a value for Rho is calculated by first solving for the Z value. The Z value is then used to look up R value and the R value is used to calculate Rho.

$$\begin{cases} Z_{0.5} = \frac{X_{Mid} - X_{Worst}}{X_{Best} - X_{Worst}} \\ R = Lookup(Z_{0.5}) \\ Rho = R * |X_{Best} - X_{Worst}| \end{cases}$$

Table 3.29: Availability SDVF

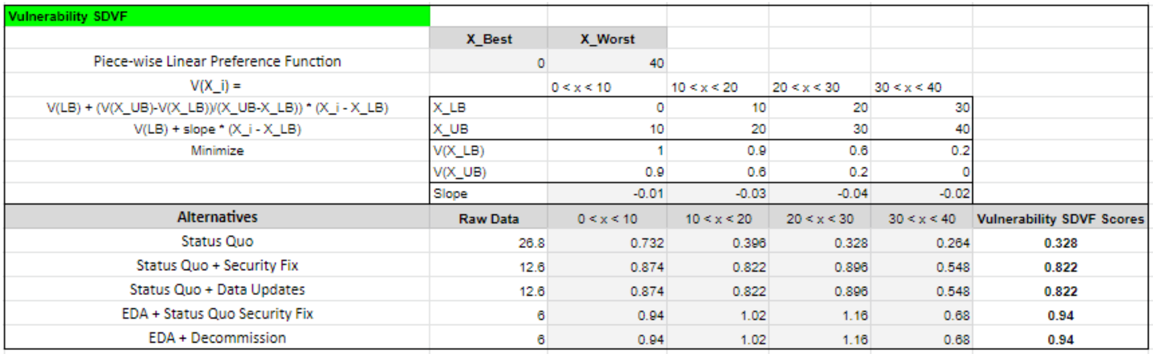


Availability SDVF			
Decreasing Exponential Preference Function	X_Best	X_Worst	Midpoint
V(X_i) = (1-exp((X_w - X_i)/Rho))/(1-exp((X_w - X_B)/Rho))	100.00%	80.00%	97.00%
Maximize	z_0.5	R Value Lookup	Rho
	0.85	-0.22	-0.044
Alternatives	**Raw Data**	**Availability SDVF Score**	
Status Quo	95.55%	0.3572971472	
Status Quo + Security Fix	95.55%	0.3572971472	
Status Quo + Data Updates	95.55%	0.3572971472	
EDA + Status Quo Security Fix	99.99%	0.9974763328	
EDA + Decommission	99.99%	0.9974763328	

Availability SDVF The Availability SDVF is a decreasing exponential preference function. The decision maker selected a midpoint of 97%. The range of values is between

100% and 80%, where V(100%) = 1 is the best value and V(80%) = 0 is the worst value. Rho was calculated to be -0.044 using the Rho calculation method. Table 3.29 shows how the alternatives were evaluated against the Availability SDVF in excel.

Table 3.30: Vulnerability SDVF



Vulnerability SDVF						
	X_Best	X_Worst				
Piece-wise Linear Preference Function	0	40				
V(X_i) =		0 < x < 10	10 < x < 20	20 < x < 30	30 < x < 40	
V(LB) + (V(X_UB)-V(X_LB))/(X_UB-X_LB)) * (X_i - X_LB)	X_LB	0	10	20	30	
V(LB) + slope * (X_i - X_LB)	X_UB	10	20	30	40	
Minimize	V(X_LB)	1	0.9	0.6	0.2	
	V(X_UB)	0.9	0.6	0.2	0	
	Slope	-0.01	-0.03	-0.04	-0.02	
Alternatives	**Raw Data**	0 < x < 10	10 < x < 20	20 < x < 30	30 < x < 40	**Vulnerability SDVF Scores**
Status Quo	26.8	0.732	0.396	0.328	0.264	**0.328**
Status Quo + Security Fix	12.6	0.874	0.822	0.896	0.548	**0.822**
Status Quo + Data Updates	12.6	0.874	0.822	0.896	0.548	**0.822**
EDA + Status Quo Security Fix	6	0.94	1.02	1.16	0.68	**0.94**
EDA + Decommission	6	0.94	1.02	1.16	0.68	**0.94**

Vulnerability SDVF The Vulnerability SDVF is a Piecewise Linear Preference Function. The decision maker selected breakpoints for the input ranges and provided a preference value for each of them. Each range then has a slope that's calculated for that range.

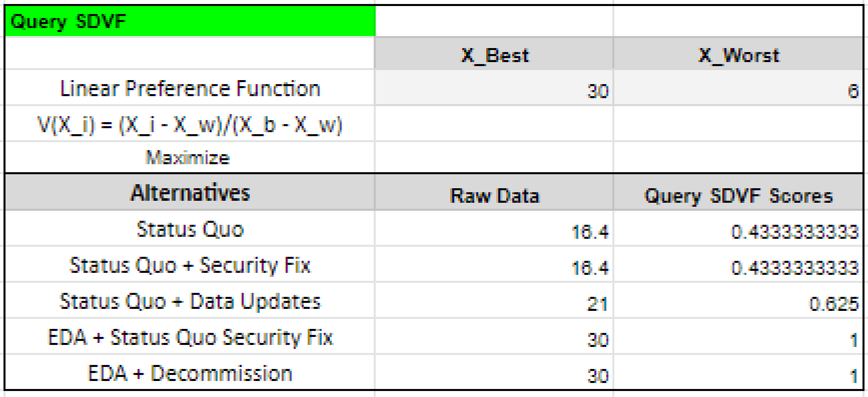
$$Range, Values, Slope = \begin{cases} 0 \leq x \leq 10 & V(0) = 1, V(10) = .9, \text{ slope} = -0.01 \\ 10 \leq x \leq 20 & V(10) = .9, V(20) = .6, \text{ slope} = -0.03 \\ 20 \leq x \leq 30 & V(20) = .6, V(30) = .2, \text{ slope} = -0.04 \\ 30 \leq x \leq 40 & V(30) = .2, V(40) = 0, \text{ slope} = -0.02 \end{cases}$$

The linear SDVF function below is calculated using the slope, lower bound x-value, and the lower bound y-value of the range.

$$V(X_i) = (X_i - X_{LB}) * Slope + V(X_{LB})$$

Table 3.30 shows how this is calculated in excel. All the linear values are calculated even if the value is not in that range. There is an “If Statement” used to select which value is the correct SDVF value to use for the final result based on whether it falls in the range.

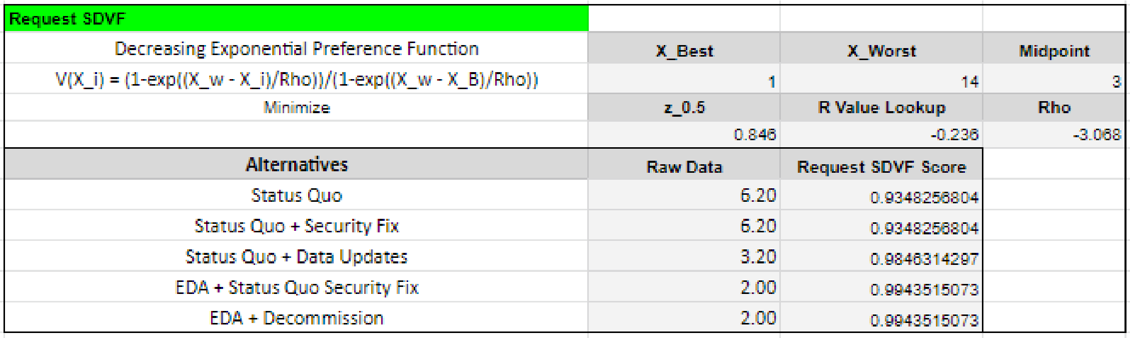
Table 3.31: Query SDVF



Query SDVF		
	X_Best	X_Worst
Linear Preference Function	30	6
V(X_i) = (X_i - X_w)/(X_b - X_w)		
Maximize		
Alternatives	Raw Data	Query SDVF Scores
Status Quo	16.4	0.4333333333
Status Quo + Security Fix	16.4	0.4333333333
Status Quo + Data Updates	21	0.625
EDA + Status Quo Security Fix	30	1
EDA + Decommission	30	1

Query SDVF The Query SDVF is a Linear Preference Function. The linear function is formed using the range of values is between 30 and 6, where V(30) = 1 is the best value and V(6) = 0 is the worst value. Table 3.31 shows how the alternative scores were calculated in excel.

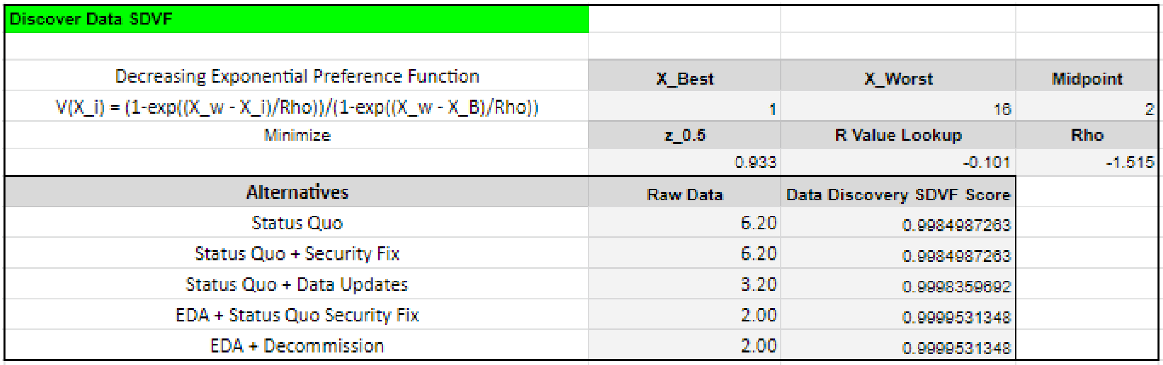
Table 3.32: Request SDVF



Request SDVF			
Decreasing Exponential Preference Function	X_Best	X_Worst	Midpoint
V(X_i) = (1-exp((X_w - X_i)/Rho))/(1-exp((X_w - X_B)/Rho))	1	14	3
Minimize	z_0.5	R Value Lookup	Rho
	0.846	-0.236	-3.068
Alternatives	Raw Data	Request SDVF Score	
Status Quo	6.20	0.9348256804	
Status Quo + Security Fix	6.20	0.9348256804	
Status Quo + Data Updates	3.20	0.9846314297	
EDA + Status Quo Security Fix	2.00	0.9943515073	
EDA + Decommission	2.00	0.9943515073	

Request SDVF The Request Data Access SDVF is a decreasing exponential preference function. The decision maker selected a midpoint of 3. The range of values is between 1 and 14, where V(1) = 1 is the best value and V(14) = 0 is the worst value. Rho was calculated to be -3.068 using the Rho calculation method. Table 3.32 shows how the alternatives were evaluated against the Request Data Access SDVF in excel.

Table 3.33: Discover Data SDVF



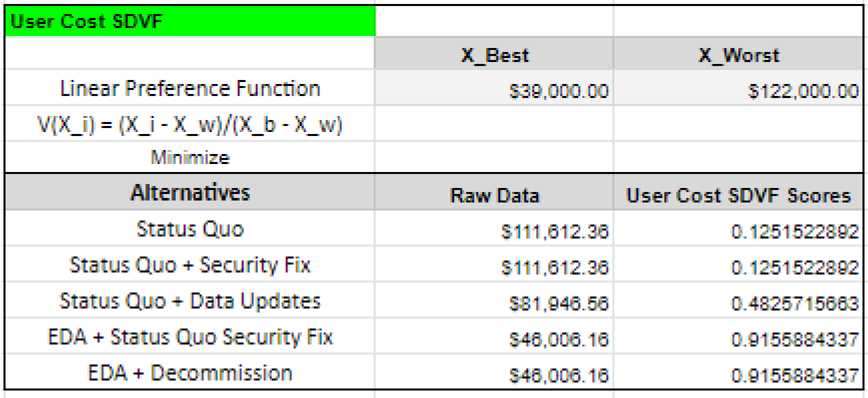
Discover Data SDVF			
Decreasing Exponential Preference Function	X_Best	X_Worst	Midpoint
V(X_i) = (1-exp((X_w - X_i)/Rho))/(1-exp((X_w - X_B)/Rho))	1	16	2
Minimize	z_0.5	R Value Lookup	Rho
	0.933	-0.101	-1.515
Alternatives	Raw Data	Data Discovery SDVF Score	
Status Quo	6.20	0.9984987263	
Status Quo + Security Fix	6.20	0.9984987263	
Status Quo + Data Updates	3.20	0.9998359692	
EDA + Status Quo Security Fix	2.00	0.9999531348	
EDA + Decommission	2.00	0.9999531348	

Data Discovery SDVF The Data Discovery SDVF is a decreasing exponential preference function. The decision maker selected a midpoint of 2. The range of values is between 1 and 16, where V(1) = 1 is the best value and V(16) = 0 is the worst value. Rho was calculated to be -1.515 using the Rho calculation method. Table 3.33 shows how the alternatives were evaluated against the Discover Data SDVF in excel.

User Cost SDVF The User Cost SDVF is a Linear Preference Function. The linear function is formed using the range of values is between $39,000 and $122,000, where V($39,000) = 1 is the best value and V($122,000) = 0 is the worst value. Table 3.34 shows how the alternative scores were calculated in excel.

System Cost SDVF The User Cost SDVF is a Linear Preference Function. The linear function is formed using the range of values is between $793,204.51 and $16,000,000,

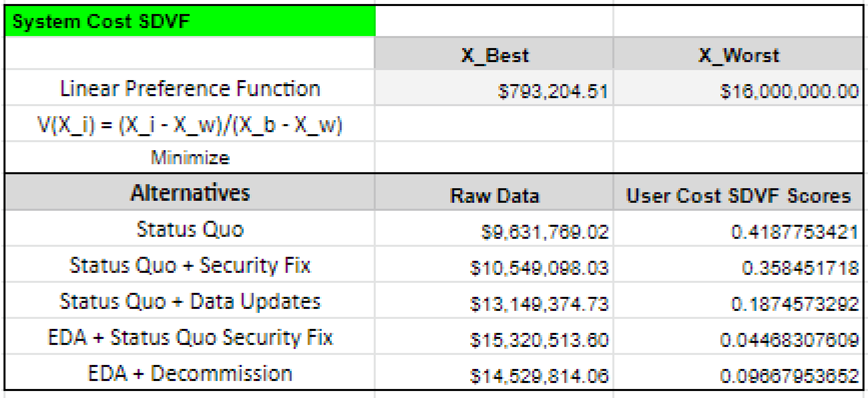
Table 3.34: User Cost SDVF



User Cost SDVF		
	X_Best	X_Worst
Linear Preference Function	$39,000.00	$122,000.00
V(X_i) = (X_i - X_w)/(X_b - X_w)		
Minimize		
Alternatives	Raw Data	User Cost SDVF Scores
Status Quo	$111,612.36	0.1251522892
Status Quo + Security Fix	$111,612.36	0.1251522892
Status Quo + Data Updates	$81,946.56	0.4825715663
EDA + Status Quo Security Fix	$46,006.16	0.9155884337
EDA + Decommission	$46,006.16	0.9155884337

where V($793,204.51) = 1 is the best value and V($16,000,000) = 0 is the worst value. Table 3.35 shows how the alternative scores were calculated in excel.

Table 3.35: System Cost SDVF

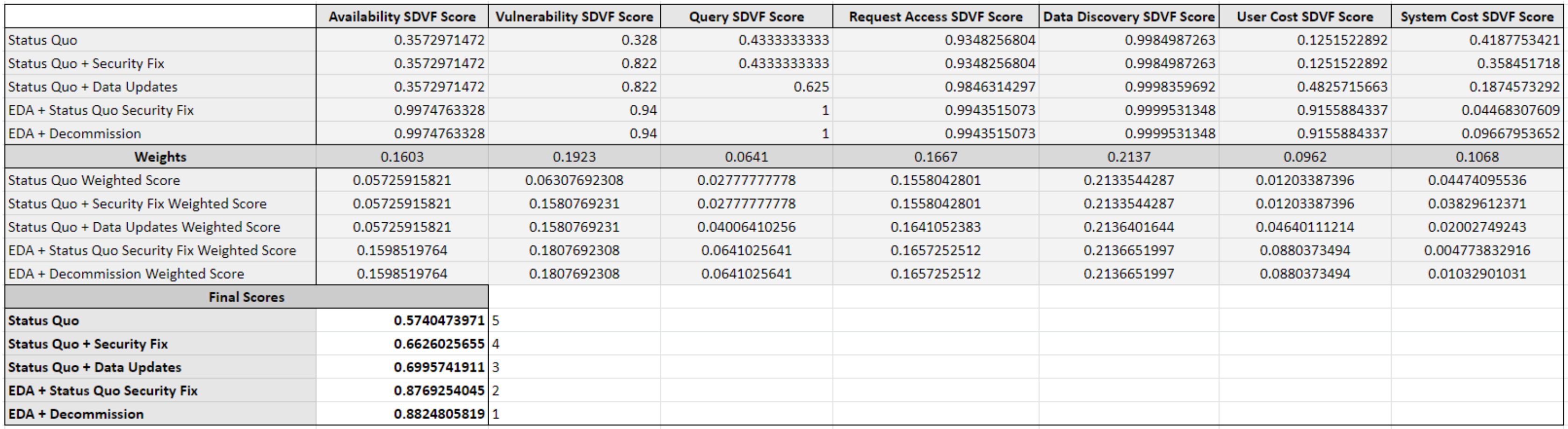


System Cost SDVF		
	X_Best	X_Worst
Linear Preference Function	$793,204.51	$16,000,000.00
V(X_i) = (X_i - X_w)/(X_b - X_w)		
Minimize		
Alternatives	Raw Data	User Cost SDVF Scores
Status Quo	$9,631,769.02	0.4187753421
Status Quo + Security Fix	$10,549,098.03	0.358451718
Status Quo + Data Updates	$13,149,374.73	0.1874573292
EDA + Status Quo Security Fix	$15,320,513.60	0.04468307609
EDA + Decommission	$14,529,814.06	0.09667953652

Value Function Each of the SDVF scores for the alternatives are summarized at the top of table 3.36. The weights for each of the objectives are listed below the SDVF scores. The weighted scores for each of the alternatives is calculated below that by multiplying the weights and the SDVF scores. The final value function scores are the sum of the weighted

SDVF scores for each of the alternatives.

Table 3.36: Final Value Function Scores



	Availability SDVF Score	Vulnerability SDVF Score	Query SDVF Score	Request Access SDVF Score	Data Discovery SDVF Score	User Cost SDVF Score	System Cost SDVF Score
Status Quo	0.3572971472	0.328	0.4333333333	0.9348256804	0.9984987263	0.1251522892	0.4187753421
Status Quo + Security Fix	0.3572971472	0.822	0.4333333333	0.9348256804	0.9984987263	0.1251522892	0.358451718
Status Quo + Data Updates	0.3572971472	0.822	0.625	0.9846314297	0.9998359692	0.4825715663	0.1874573292
EDA + Status Quo Security Fix	0.9974763328	0.94	1	0.9943515073	0.9999531348	0.9155884337	0.04468307609
EDA + Decommission	0.9974763328	0.94	1	0.9943515073	0.9999531348	0.9155884337	0.09667953652
Weights	0.1603	0.1923	0.0641	0.1667	0.2137	0.0962	0.1068
Status Quo Weighted Score	0.05725915821	0.06307692308	0.02777777778	0.1558042801	0.2133544287	0.01203387396	0.04474095536
Status Quo + Security Fix Weighted Score	0.05725915821	0.1580769231	0.02777777778	0.1558042801	0.2133544287	0.01203387396	0.03829612371
Status Quo + Data Updates Weighted Score	0.05725915821	0.1580769231	0.04006410256	0.1641052383	0.2136401644	0.04640111214	0.02002749243
EDA + Status Quo Security Fix Weighted Score	0.1598519764	0.1807692308	0.0641025641	0.1657252512	0.2136651997	0.0880373494	0.004773832916
EDA + Decommission Weighted Score	0.1598519764	0.1807692308	0.0641025641	0.1657252512	0.2136651997	0.0880373494	0.01032901031
Final Scores							
Status Quo	**0.5740473971**	5					
Status Quo + Security Fix	**0.6626025655**	4					
Status Quo + Data Updates	**0.6995741911**	3					
EDA + Status Quo Security Fix	**0.8769254045**	2					
EDA + Decommission	**0.8824805819**	1					

3.4 Framework Results

Overview of Results Results can be presented to the decision maker in their preferred format, whether it's a PowerPoint or the excel sheet itself. Summarizing the work done in order to achieve these results is important. The decision maker should have an understanding of how the data was gathered and analyzed. The decision maker should be confident that the model is accurate.

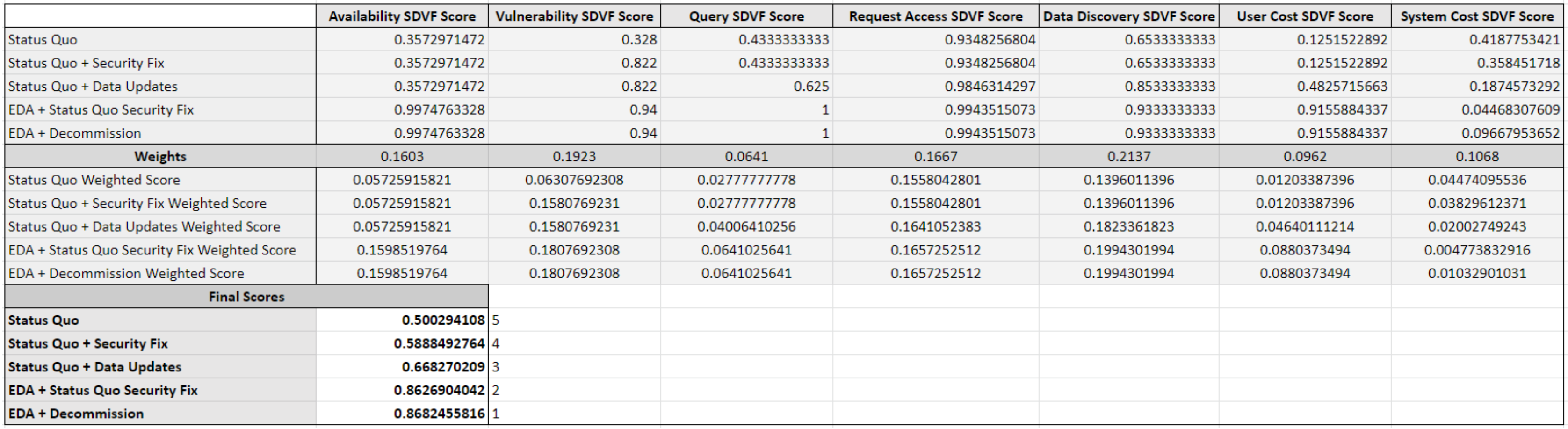
Some coaching may be required for a decision maker to understand what questions they can ask and get immediate responses to and which questions will need additional research. This section will provide examples of how the models can help answer questions a decision maker has.

3.4.1 Initial Decision Results

The completed model in section 3.3.8 has the final ranking of all the alternatives. It includes how each of the alternatives scored for each of the objectives and the weights for each of the objectives. During the presentation of the initial results it would be good to point out to leadership what the driving values are behind the ranking.

This might lead to the decision maker rethinking the weights of their objectives or how their preference changes over the different range of scores. For example there is not much variability in the score for Data Discovery and it's the highest ranked objective which makes it a key driver in the results. These scores were determined using a decreasing exponential preference function. Changing the decreasing exponential to a linear preference function will change how the raw scores are normalized and provide greater variability in the final results.

Table 3.37: Final Scores Data Discovery Modification



	Availability SDVF Score	Vulnerability SDVF Score	Query SDVF Score	Request Access SDVF Score	Data Discovery SDVF Score	User Cost SDVF Score	System Cost SDVF Score
Status Quo	0.3572971472	0.328	0.4333333333	0.9348256804	0.6533333333	0.1251522892	0.4187753421
Status Quo + Security Fix	0.3572971472	0.822	0.4333333333	0.9348256804	0.6533333333	0.1251522892	0.358451718
Status Quo + Data Updates	0.3572971472	0.822	0.625	0.9846314297	0.8533333333	0.4825715663	0.1874573292
EDA + Status Quo Security Fix	0.9974763328	0.94	1	0.9943515073	0.9333333333	0.9155884337	0.04468307609
EDA + Decommission	0.9974763328	0.94	1	0.9943515073	0.9333333333	0.9155884337	0.09667953652
Weights	0.1603	0.1923	0.0641	0.1667	0.2137	0.0962	0.1068
Status Quo Weighted Score	0.05725915821	0.06307692308	0.02777777778	0.1558042801	0.1396011396	0.01203387396	0.04474095536
Status Quo + Security Fix Weighted Score	0.05725915821	0.1580769231	0.02777777778	0.1558042801	0.1396011396	0.01203387396	0.03829612371
Status Quo + Data Updates Weighted Score	0.05725915821	0.1580769231	0.04006410256	0.1641052383	0.1823361823	0.04640111214	0.02002749243
EDA + Status Quo Security Fix Weighted Score	0.1598519764	0.1807692308	0.0641025641	0.1657252512	0.1994301994	0.0880373494	0.004773832916
EDA + Decommission Weighted Score	0.1598519764	0.1807692308	0.0641025641	0.1657252512	0.1994301994	0.0880373494	0.01032901031
Final Scores							
Status Quo	**0.500294108**	5					
Status Quo + Security Fix	**0.5888492764**	4					
Status Quo + Data Updates	**0.668270209**	3					
EDA + Status Quo Security Fix	**0.8626904042**	2					
EDA + Decommission	**0.8682455816**	1					

The results of switching the preference function from an exponential to a linear for the data discovery objective are shown in table 3.37. The final ranking of the results did not change in this case, but the spread between the best and worst value ranking has increased, which may be valuable in future discussion with the decision maker. This is an example of looking at how the different preference functions can impact the model and final results. The goal is to make sure the model reflects the decision maker's preferences. These modifications can be quickly made on the fly.

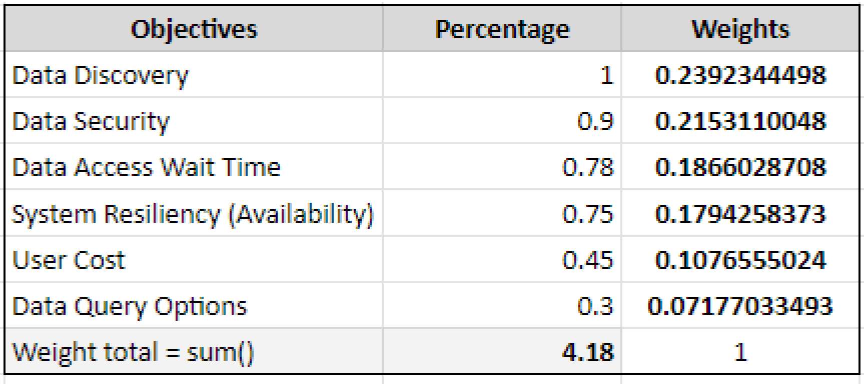
Immediate Response What-If Questions There are several questions the model could be modified to answer during discussions with the decision maker, similar to the previous example. Decision maker's are likely to have many "what if" questions. Here are a list of questions the model could be quickly modified to answer right away.

- What if I re-prioritized my ranking of the objectives? How would that change the results?
- What if I changed the midpoints, ranges, or type of preference function?
- What if one of the alternatives improved their score in an area? How would that change the results?
- What if we removed an objective all together? How would the results rank?
- If I wanted a solution built by X time frame, which would be the best one?

Having a model that can quickly answer these kinds of questions will help the decision maker better understand what the model is currently capable of. It helps the decision maker better understand what the trade-offs are and which alternative provides the organization the best value, based on their objectives.

Cost Value Model Modification For the current decision model, one of the objectives includes total cost of the system. In practice this objective is likely to dependant on the other objectives, which can bias the model. A quick change can be made to remove cost as an objective from the model. The weights of the rest of the model can quickly be recalculated, as shown in table 3.38.

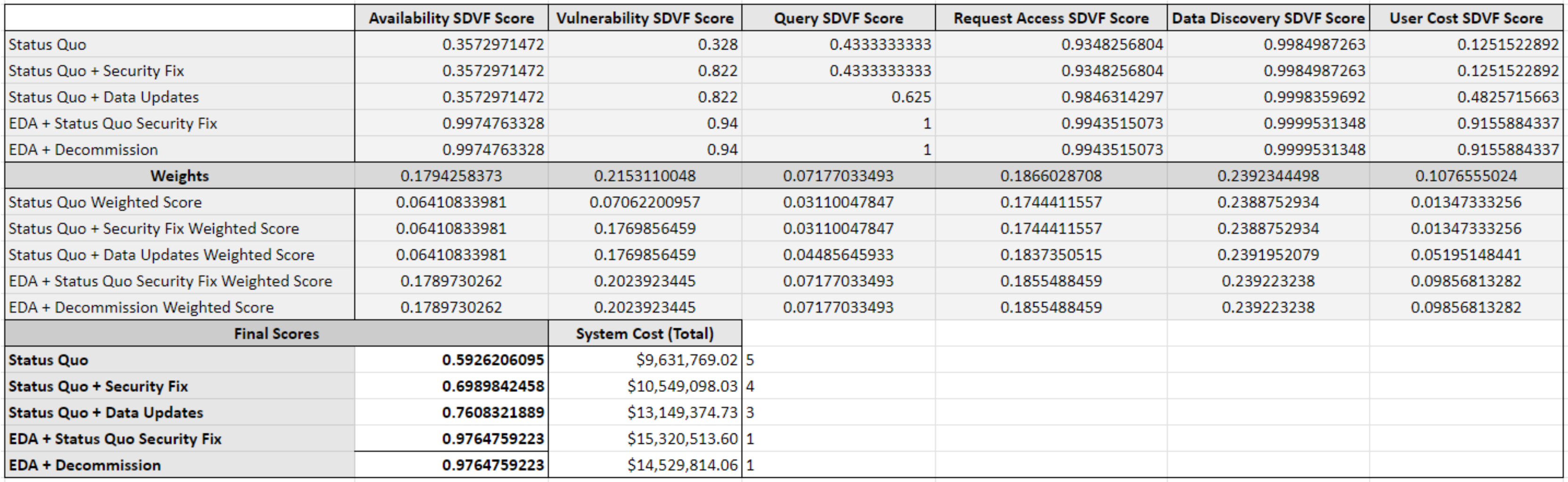
Table 3.38: Updated Weights



Objectives	Percentage	Weights
Data Discovery	1	**0.2392344498**
Data Security	0.9	**0.2153110048**
Data Access Wait Time	0.78	**0.1866028708**
System Resiliency (Availability)	0.75	**0.1794258373**
User Cost	0.45	**0.1076555024**
Data Query Options	0.3	**0.07177033493**
Weight total = sum()	**4.18**	1

The with the new weights and the system cost objective removed the new ranking can be viewed. There are some changes in these results. The EDA + Status Quo Security Fix and EDA + Decommission alternatives are both ranked number 1. This is because they are they provide the same value to the organization, the only difference was cost of the decommissioned data system. By breaking out the cost from the value, the results can now be graphed which provides the decision maker another view. Figure 3.48 shows the total cost of the system over 85 months and the value that alternative brings. When looking at the cost and value graph, you want to see results with low costs and high values. Those would be located on the bottom right of the graph. For this example there are no alternatives in the bottom right, but there are alternatives in the upper right, which means there are high value and high cost alternatives.

Table 3.39: Updated Value Function without System Cost



	Availability SDVF Score	Vulnerability SDVF Score	Query SDVF Score	Request Access SDVF Score	Data Discovery SDVF Score	User Cost SDVF Score
Status Quo	0.3572971472	0.328	0.4333333333	0.9348256804	0.9984987263	0.1251522892
Status Quo + Security Fix	0.3572971472	0.822	0.4333333333	0.9348256804	0.9984987263	0.1251522892
Status Quo + Data Updates	0.3572971472	0.822	0.625	0.9846314297	0.9998359692	0.4825715663
EDA + Status Quo Security Fix	0.9974763328	0.94	1	0.9943515073	0.9999531348	0.9155884337
EDA + Decommission	0.9974763328	0.94	1	0.9943515073	0.9999531348	0.9155884337
Weights	0.1794258373	0.2153110048	0.07177033493	0.1866028708	0.2392344498	0.1076555024
Status Quo Weighted Score	0.06410833981	0.07062200957	0.03110047847	0.1744411557	0.2388752934	0.01347333256
Status Quo + Security Fix Weighted Score	0.06410833981	0.1769856459	0.03110047847	0.1744411557	0.2388752934	0.01347333256
Status Quo + Data Updates Weighted Score	0.06410833981	0.1769856459	0.04485645933	0.1837350515	0.2391952079	0.05195148441
EDA + Status Quo Security Fix Weighted Score	0.1789730262	0.2023923445	0.07177033493	0.1855488459	0.239223238	0.09856813282
EDA + Decommission Weighted Score	0.1789730262	0.2023923445	0.07177033493	0.1855488459	0.239223238	0.09856813282
Final Scores		**System Cost (Total)**				
Status Quo	**0.5926206095**	$9,631,769.02	5			
Status Quo + Security Fix	**0.6989842458**	$10,549,098.03	4			
Status Quo + Data Updates	**0.7608321889**	$13,149,374.73	3			
EDA + Status Quo Security Fix	**0.9764759223**	$15,320,513.60	1			
EDA + Decommission	**0.9764759223**	$14,529,814.06	1			

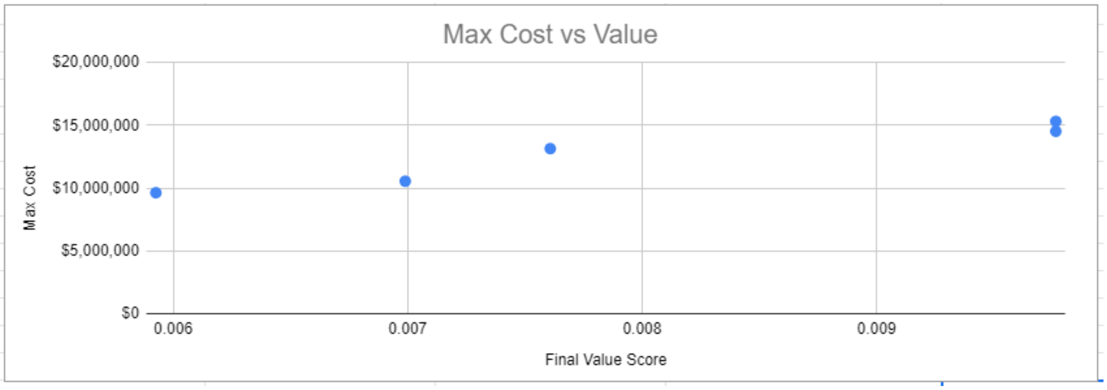


Figure 3.48: Max Cost vs Value

This cost value is based on the total cost after 85 months so it takes into account the O&M after almost 5 years. If the decision maker were interested in seeing how the value and schedule for each of the alternatives, that could also easily be done. Instead of using the O&M timeline to estimate the total cost, the development timeline would be used to estimate the total cost.

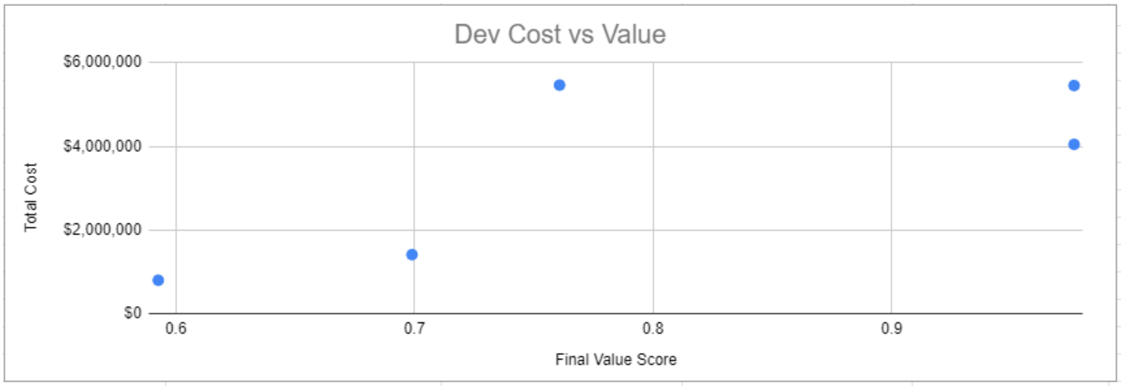


Figure 3.49: Dev Cost vs Value

Figure 3.49 shows the new cost and value relationship based on the development timeline. So it shows how much each alternative would cost for development in relation to the value it brings.

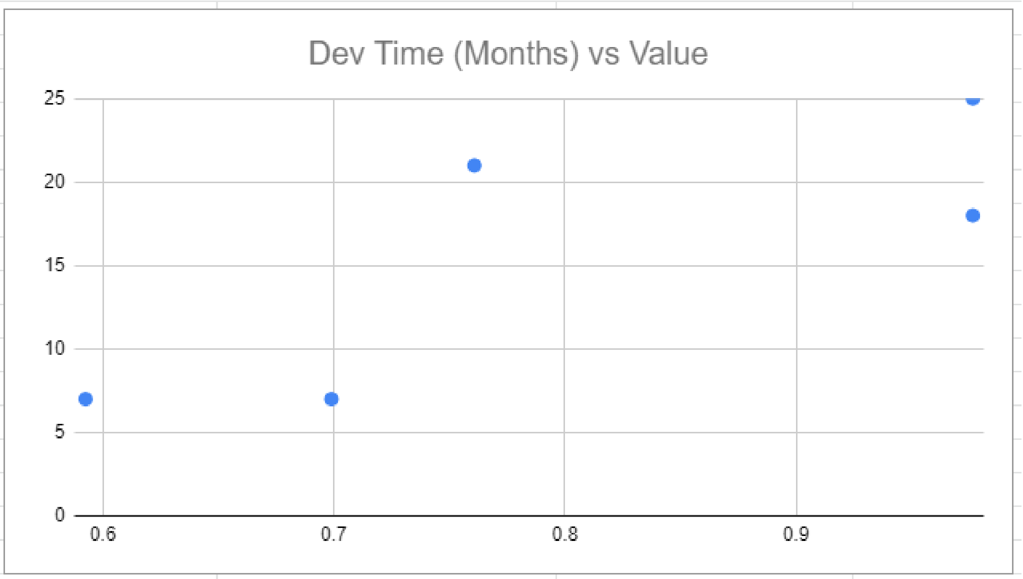


Figure 3.50: Dev Time vs Value

Figure 3.50 shows the value of each alternative in relation to the time it takes to develop it. This can be useful when leadership is concerned about how long it will take to deliver a capability. When looking at the Time vs Value graph, the results you are hoping for are fast delivery and high value. Those would be located in the bottom right of the graph. Bad options would be an alternative that is slow to deliver and low in value, those would be located in the upper left corner of the graph. For this example there are not alternatives in those areas, the results fall somewhere in between. There are alternatives that deliver quickly but provide low value. There are alternatives that deliver slowly, but provide high value.

The final view that can be created is shown in figure 3.51. This shows the relationship between cost of development and length of time to complete the development. Because cost is dependant on the length of time to develop these will have close to a linear relationship. High Cost usually means longer delivery time. There could be other factors that drive up

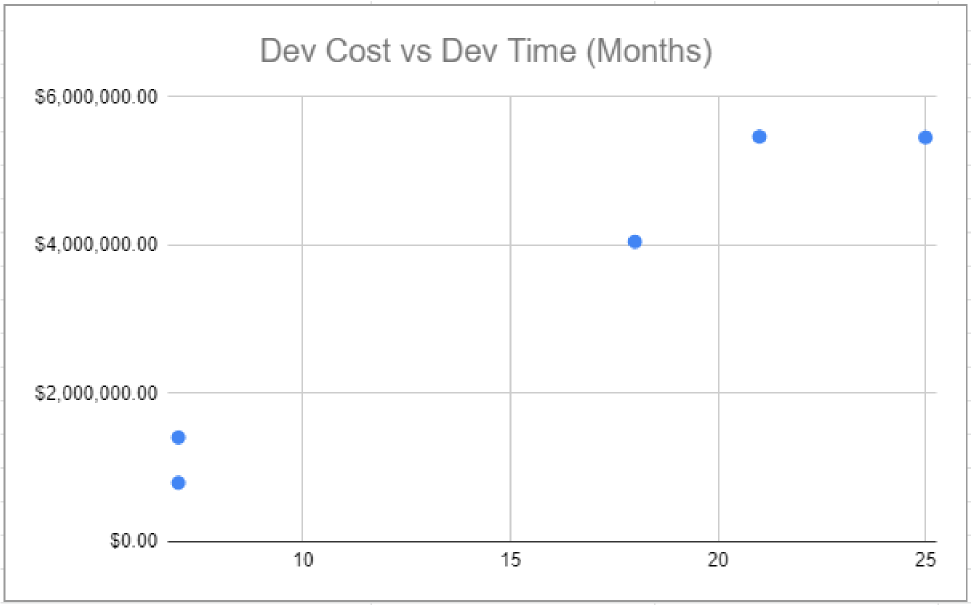


Figure 3.51: Dev Cost vs Dev Time

costs, like licensed products, so it is still useful to compare alternatives this way. The results you are hoping for are the low cost and quick delivery time, but that tells you nothing about the value the system brings, which is why the other graphs are important.

3.4.2 Decision Maker Final Decision and Future Discussions

The model should be modified as needed to provide the decision maker the data they need in order to make a final decision on an architecture. Some of the questions they need answered will take some additional data gathering to answer. Having the model helps to figure out what information is needed to answer the Decision Maker's question. The data gathering can be more targeted and happen much more quickly.

Swift Response What-If Questions Here are a list of questions a Decision Maker will have that require additional data gathering for the model.

1. What if we am interested in a commercial off the shelf (COTS) solution for this

architecture? How would that alternative rank?

2. What if we wanted to support a new objective for the organization? How would that effect the value, cost, and schedule?

3. What if we wanted to decommission more systems? How would that effect the results?

4. What if we wanted to add another data system? How would that effect things?

For questions 1, 3, and 4 they represent additional alternatives that can be evaluated and ranked with the other alternatives. But because the model and evaluation tests have already been built the data gathering should go quickly. A questionnaire can be created based on the model to gather all the information needed in order to populate the models. It should include questions to gather information need for the value assessment and the cost model. So when any new alternative is brought up and needs to be evaluated, the task of gathering the information needed can be delegated to multiple people or covered in a few meetings rather than spending months researching.

For question's like number 2, a new objective or a scope increase a new evaluation test needs to be created. Once the test has been developed information can be gathered on each of the alternatives for that specific test. The information can the be incorporated into the value and cost models. Using the model for assessing the scope change can show the decision maker how much value increase the scope provides and what the impacts are on cost and schedule. This can help the decision maker determine whether the scope increase provides enough value to warrant the cost. This can also be useful when trying to budget for future requirements to the project. If the scope increase is out of the budget, knowing the additional funding required to support the new requirements means it can be budgeted for in the future.

Decision Support during Development Even after a decision on an architecture is made, the model can be used to support questions as they arise during the development process. New alternative solutions may appear and the model can be used to assess them quickly without having to stop the development team from working. It can help the team make data driven decisions on changes. The development team should not stop building towards their current architecture plan unless the assessment of a new alternative reveals significant value increase, with minimal cost. This can help teams be less reactionary and stay focused on delivery.

New requirements will come up and the model can be used to determine whether the requirements will take the project out of budget before work begins on the new requirement. There might be trade-offs that need to be made because this new requirement is more important now than other objectives. Having the model will help the decision maker figure out what objectives to drop in order to support the new one. Having the model will help determine timeline delays the new requirement will have on the project.

The model is a living artifact that can be agile and modified as needed throughout the project to help the decision maker make data driven decisions. It can be used to support the development team by minimizing scope changes. It can be used to support decision makers by helping to develop cost plans and schedules for future capabilities. It helps the organization by making sure the objectives of the system are aligned with their strategic goals. It is useful for the initial decision and future decisions that need to be made.

Chapter 4: Evaluation

Overview This chapter will demonstrate how the model-based decision framework improves the decision making process for determining an enterprise data architecture. First section will explain the goals for the evaluation and experiment. The next section will cover what the survey experiment was and how it was set up to answer the evaluation questions. The survey results are then presented in the following section. The final section of this chapter goes over the conclusions that can be drawn from the survey results.

4.1 Evaluation Overview

Evaluation Goals There are three goals for this thesis evaluation. The first goal is to validate the problem claims. Do development teams have challenges making decisions on an enterprise data architecture design? Do they have trouble justifying and backing up their decisions? Are they able to respond to the question's their organization's leadership will have about the project? Do the development methods they use to build the architecture provide them enough information to successfully build their enterprise data architecture within their budget and schedule? To satisfy this goal, the survey will need to gather information on whether or not development teams encounter these kinds of scenarios.

The next goal is to then understand the impact these problems have on a development teams ability to successfully develop an enterprise data architecture. If a development team does encounter a situation where their project is being questioned, how were they able to respond? Do the development methods they use to build the architecture provide them enough information to quickly respond to leadership questions about the project? To satisfy this goal, the survey will need to gather information on different possible ways a project could be impacted.

And the last goal is to determine if the solution this thesis proposes would help improve the development teams ability to respond to these problems and minimize the impact. Would the whole model or components of the model improve the development teams ability to respond effectively. To satisfy this goal, the survey needs to include a question about whether or not the model would have been useful to the development team.

Unofficial goals would be to see if there are any patterns revealed in the responses. For example if an smaller organizations are more likely to succeed in building an enterprise data architecture. Or which development processes are most commonly used to develop an enterprise data architecture. Interesting patterns would be helpful with future research.

4.2 Survey Experiment

Survey Goals The goals of the survey are to gather data to answer the evaluation goals. The survey was developed to capture information from people who worked on data enterprise architecture projects. The key questions we want the survey's data to reveal.

1. What development process did they use to develop the enterprise data architecture?
2. Did the development teams struggle to make a decision on their final architecture?
3. Did development teams encounter these problem scenarios?
4. Were the development team prepared to respond to those scenarios?
5. How was the project impacted by the scenarios?
6. Would the decision model have helped reduce the impact, in their opinion?

4.2.1 Survey Setup

Survey Overview The survey was conducted for a week in Oct 2021 for a week. It was communicated out over LinkedIn and by word of mouth. There were a total of 30 questions asked. To simplify the analysis, the majority of the questions were multiple choice or checkboxes. The survey did have an other option where someone could write in an option that wasn't available. Only two questions had free text options.

The survey was broken into three parts. The first part of the survey was to gather information about the person responding to the survey and the organization the enterprise data architecture was developed for. The next part of the survey was gathering information about the enterprise data architecture project itself. The final part of the survey was about the different scenarios a enterprise data project might encounter.

4.2.2 Survey Organization and Role Questions

This section of the survey was to understand who the person responding to the survey was and the type of organization they worked for. These are the questions that were asked.

1. Have you worked on developing an enterprise data architecture?
2. What type of organization was the enterprise data architecture for?
3. How large is the organization?
4. What was your role in developing the enterprise data architecture?
5. What kind of impact on the decision making did you have?

The first question was a prerequisite question. If the response was no then the survey skipped to the end. The type of organization, question 2, was asked to gather information on what industries are currently building enterprise data architectures and maybe identify patterns in the problem scenarios. Asking about the size of the organization was also to see if there were patterns in the size of the organization and the challenges they encounter. The last two questions were to understand the perspective of the person filling out the survey.

4.2.3 Survey Project Context Questions

This section of the survey was to understand information about the enterprise data architecture project itself. These are the questions that were asked.

1. Which development process was used to develop the enterprise data architecture?
2. How long did it take to determine what the final architecture would be?
3. How many times did the architecture change?
4. Did the data architecture get implemented into completion?

Question 1 provided multiple choice options for development processes to be selected, Agile, Waterfall, V-Model, and Acquisition for a commercial capability. Another option was provided in case there was another development process used. Question 2 is meant to get an understanding of how long it takes to make a final decision on an enterprise data architecture. The intent behind Question 3 is to understand how many times a team made a decision and changed it. Question 4 is mean to give context to how many responses are still in the process of building their enterprise data architecture. It is also used to determine how many were successful and how many failed.

4.2.4 Survey Scenario Questions

There were four scenarios presented in the survey. The following questions were asked for each scenario.

1. Did you experience a scenario similar to this?
2. How long did it take to respond?
3. How did it impact your project?
4. Did you have the information on hand to respond or was research required?

Question 1 is a prerequisite question and is used to determine if enterprise data architecture development teams encounter these kinds of scenarios. Question 2 is used to understand the length of time it took for the team to respond. This response time is where using the decision model could help reduce. Time is not the only impact a scenario can have, so this question provided checkboxes for teams to select and an other option in case there were more impacts not listed. Question 4 is trying to understand how prepared the team was for responding to the scenario. Did the development process they used help them collect the information needed in order to be prepared to respond.

Scenario Overview A scenario is meant to be a situation that a development team can encounter which leads to a broad set of questioning that can occur. These scenarios were developed because the decision model can help answer these questions with data.

Scenario #1: Alternative Solution The alternative solution scenario is for when teams are shown another potential enterprise data architecture solution. This can happen during or after they have made a decision on their enterprise data architecture solution. This is also an adapt, build, or buy decision. In the case where the team hasn't chosen an architecture yet, they can incorporate this alternative into their initial evaluation of alternatives.

In the case that the team has already made a decision on their architecture and have put time and resources against it, they would not want to pivot to this new alternative solution unless the value increase was worth the "rework costs." This is normally a tricky situation to answer and can take time away from the development team continuing to make progress. The model has the potential to calculate the value and cost differences quickly. Only one person needs to work the assessment, while the rest of the development team can continue building.

Scenario 1 was stated as follows: "The organization's leadership has been informed about an enterprise data architecture solution and want the team to review it to determine if it's an option they should pursue. The solution can be a vendor solution, another organization's solution, or another internal program to the organization. Example Question: How does this alternative measure against the current architecture approach? Please respond to the questions below."

An additional question is asked along with the other four questions. Below is the specific question for Scenario #1.

1. Would having a model for evaluating and comparing architectures have been useful?

The intent behind this question is to see if the decision model would have been helpful to team. The value assessment part of the model is what is being considered here. Does the person filling out the survey think that having a way to compare and evaluate different architecture would have helped them respond to the scenario.

Scenario #2: New Strategic Initiative The scenario can occur when an organization decides they need to make sure their programs are aligned with their strategic goals. Usually this is focused on making sure they are spending money on programs that are directed towards achieving those goals. If the project isn't aligned to those goals, then funding for the project may get cut or the program could come under more scrutiny from the organization's leadership.

Because the decision model starts off with defining what the goals are for the system and making sure they are aligned with the organization's goals, the team already has the system's objectives documented. This can help the team communicate how the system is working to support the organization's goals. The team has already thought about this scenario on a strategic level and how to achieve it on a tactical level.

If the new initiative is not covered under any of the current systems goals, then the team can think about adding a new objective and increasing the systems scope, or the team can push back and state that the new initiative is out of scope for the system. Pushing back and saying the new initiative is out of scope is less risky in this case because the team can demonstrate how they are already supporting the other organizational goals.

Scenario 2 was started as follows: "There is a change in the organization's management. The new leadership wants to make some changes to the organization and has stated new goals they want the organization to achieve. The new leadership wants to make sure the whole organization is aligned to meet those goals. Example Question: How does the enterprise data architecture project measure against the new strategic initiative(s)? Please respond to the questions below."

An additional question is asked along with the other four questions. Below is the specific question for Scenario #2.

1. Would having already developed objectives for the project with measurement criteria on how the project would achieve those objectives have been helpful?

The intent behind this question is to validate that identifying and aligning the objectives to the values of the organization would be helpful to the team.

Scenario #3: Scope Increase The scope increase scenario is when a new requirement is levied on the development team. There are different types of requirements with different levels of impact. Some are small with minimal impact on cost and time. The larger scope increases are the ones that would be worth calculating how it will impact the team's current plan. New requirements that are on the level of a new objective for the system, could potentially effect the system design, the cost, and timeline for delivery. It might even require major rework to the system.

The decision model would provide a starting place to show how the value of the system could be assessed based on the new objective. It can also estimate the cost and timeline for that change. This gives the decision maker more insight into how much of an impact the new requirement will have on the system. The benefit from the model is also being able to ask for additional funding to meet the new requirement. All this can be done while the development team continues to work before a final decision is made on whether to increase the scope or not.

Scenario 3 was stated as follows: "The organization's leadership has identified a new requirement and they think the enterprise data project can address it. The new requirement could be functional or non-functional. Example Question: How can the enterprise data architecture meet the new security requirement? Please respond to the questions below."

An additional Scenario #3.

1. Would having a value estimate and cost model which can estimate the change in scope's impact on cost and value have been helpful?

The intent behind this question is to validate whether the decision model would be useful to teams that have encountered this scenario. Does the person filling out the form think the data would have been valuable to have?

Scenario #4: Cost Projections This cost projections scenario is where the organization may be looking to plan and budget for the next several years. They might be looking at ways to cut back spending or realign resources to other initiatives. They will want to know over several years what the budget forecast is for the project and how those resources are being used. It might not just be the total cost. They may want a more detailed breakdown of the skill sets and number of people required. They might want information costs on software licenses, hardware, rental space, etc. They may want to know how the team spent the money previously and what the return on investment was.

The decision model can help address all these types of questions. It was demonstrated in Chapter 3 that the different alternatives could have their total cost estimated for 85 months or 5 years. The model can be adjusted to include things like inflation and change in skill set hourly rates. The data has already been collected and can be easily tweaked to address the scenario questions.

Scenario 4 was stated as follows: "The organization is developing plans for the next few years. They want to know how much to budget for each project over the next three years. They are looking to reduce IT costs and have identified systems they want to replace with the enterprise data architecture. Question: What will the total cost of ownership be for the organization in three years? Please respond to the questions below. "

An additional question is asked along with the other four questions. Below is the specific question for Scenario #4.

1. Would having a cost model which can estimate the development cost and O&M costs for the project have been helpful?

The intent behind this question is to understand if the decision model's cost model component would have been helpful. Does the person filling out the survey think a cost model would have been helpful to have for their project?

4.3 Survey Results

Results Overview After one week, there were 11 responses total. One response was a non-response because they selected "no" on the question asking if they had worked on an enterprise data architecture. Some of the text option fields were used so the data needed to be cleaned up for analysis.

Two respondents left comments at the end of the survey to give more context to their responses. The first respondent specified that their project had been going on 8 years and was still not an enterprise level capability. They claimed it was "largely due to lack of a well-defined, effective data architecture," which is an issue that the decision model could address. The second respondent had made a comment about cost models from their experience being inadequate and biased. They said, "Cost models historically in my experience are biased based on leadership opinions and tend to lack true cost," which is another issue the decision model could address. Working through the value-modeling process can help to reveal bias that leadership may have. The decision model provides them the opportunity to make decisions based on data and not their bias instinct. The cost models should reflect all costs including development, O&M, infrastructure, and licensing costs.

4.3.1 Survey Respondent Context Results

Organization and Role Question Results Figure 4.1 shows the results of the organization and respondent's role questions. There were 10 people who responded to the survey whom had worked on an enterprise data architecture. There were 7 large government organizations and one medium sized government organization. There were two commercial organizations one medium and one large. Half of the respondents were System Architects or designers. Two of the respondents were organizational decision makers and the other three would be classified in the area of build support for their projects. The impact on decision making was split evenly between direct impact and influence. Both of the organizational decision makers did respond that the had direct impact on the project and two of the system architects responded they only had influence.

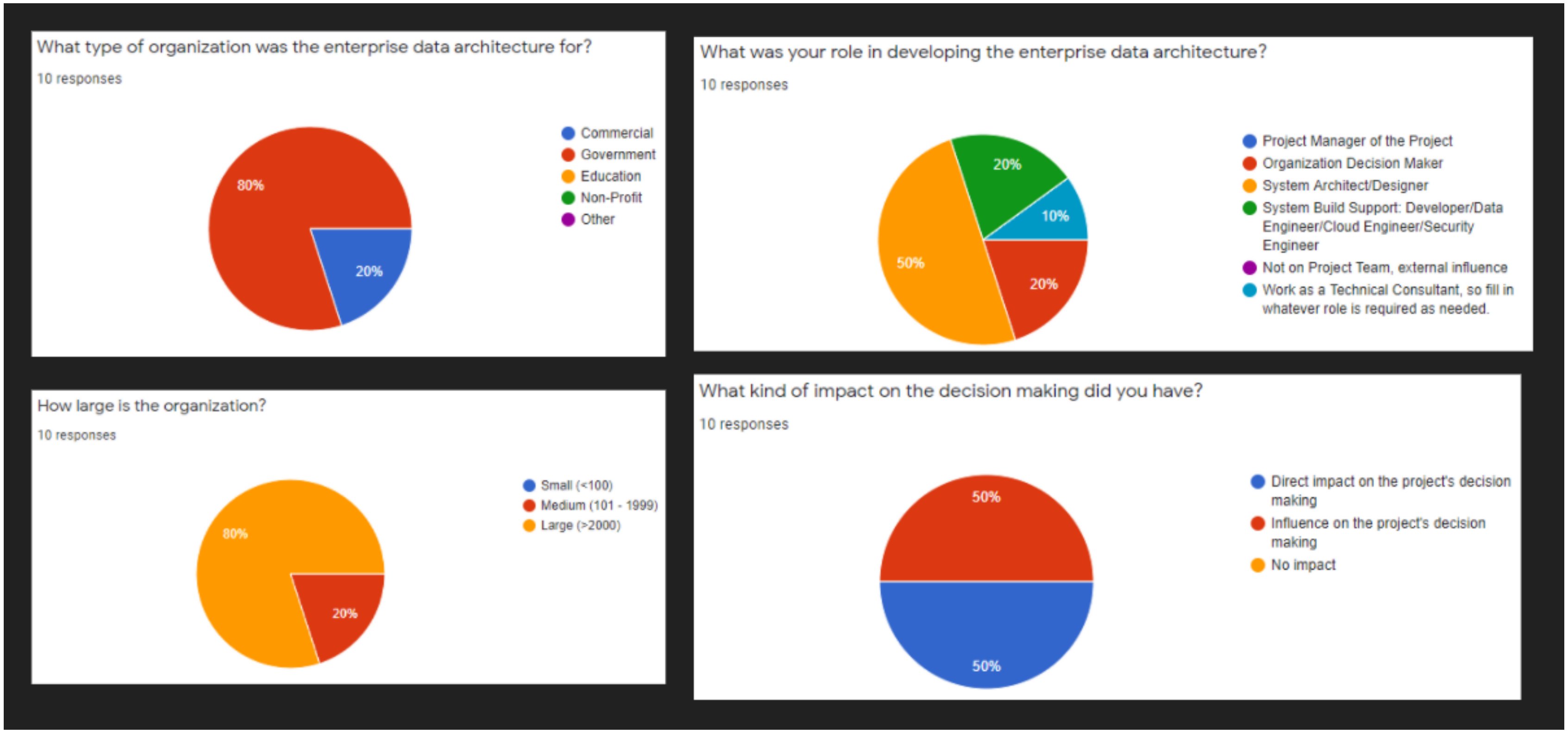


Figure 4.1: Role and Organization Survey Results

Project Context Question Results Figure 4.2 shows the results for the project context questions. Half of the enterprise data architecture projects used agile for their development process. Two projects followed a V-Model development approach and one project used waterfall. One project used a combination of acquisition, V-Model, and agile development. Agile is a popular choice for developing an enterprise data architecture.

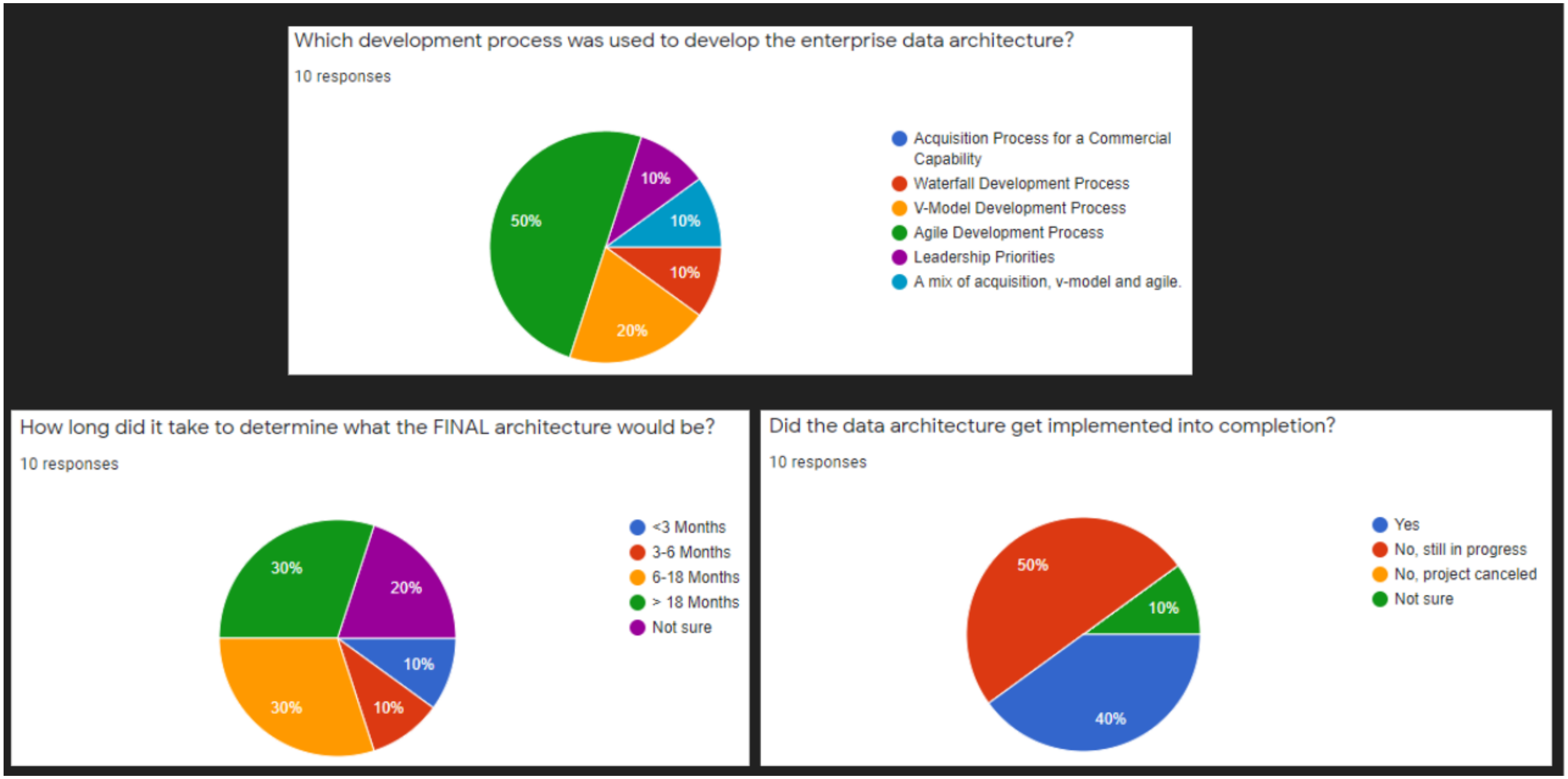


Figure 4.2: Survey Project Results

For the length of time it takes determine the final architecture the respondents had a variety of responses. Two respondents didn't know how long it took. One respondent was able to determine their final architecture in less than 3 months. One respondent was able to determine their final architecture in 3-6 months. The majority of respondents took longer the 6 months to determine their final architecture design. For three of the respondents, it took longer than 18 months which is 37.5% of those who responded after filtering out those who didn't know how long. Based on these results it's clear it takes a significant amount of time to design an enterprise data architecture.

For the two projects that were able to determine their final architecture in less than 6 months, they were both projects for medium sized organizations. Those are organizations with 101 - 1999 people in the organization. This could imply that large organizations, 2000+ people, are more likely to have challenges with determining their final enterprise data architecture design.

Half of the respondents said the enterprise data architecture projects they worked on were not completed and still actively being developed. One respondent wasn't sure of the project's status. Four of the respondents said their enterprise data architecture was completed.

The results for the "How many times did the architecture change?" question were mixed because the format of the survey question was a textbox. Six responses to this question could be categorized in 10 or less design changes. Three responses could be categorized as more than 10 design changes. One response was a not applicable. Figure 4.3 shows the results when they are bucketed into these groups.

The spread of responses was from zero changes to thirty or more changes to the architecture design. Because of the variety in responses this question might not have been defined clearly enough and was interpreted differently. The question should have stated more clearly that it was asking about significant architecture design changes. Design changes that would require different cost calculations and changes to the delivery schedule.

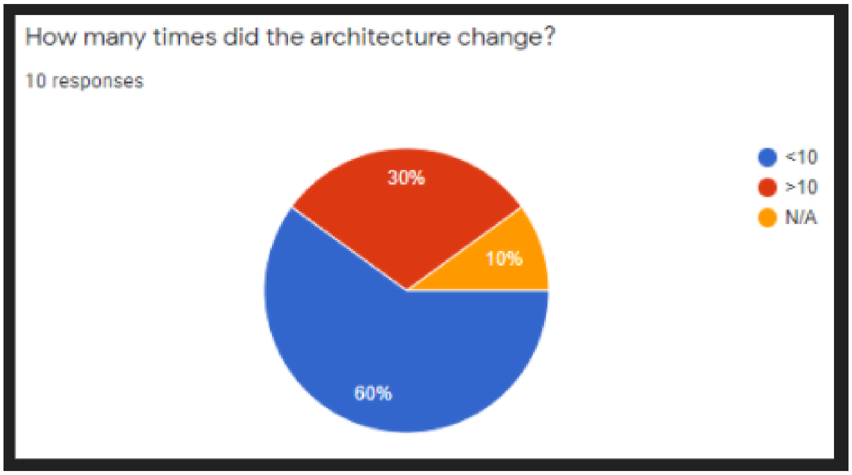


Figure 4.3: Architecture Change Survey Results

4.3.2 Scenario Question Results

Scenario #1: Alternative Solution Results Nine of the ten respondents said they had experienced Scenario #1. Figure 4.4 describes the results for the questions the nine people answered. A third of the people were able to respond in a week or less. These projects also selected that this scenario did not have any impact on their project. The project that had responded in 2-3 days was the project that also selected they had the information available. Given these results it's likely that the teams who had the more information were able to respond more quickly.

For the six projects that took longer than a month to respond, this scenario had a variety of impacts on their project. Five of the six responded that they had time delays. Four projects experienced a shift in their development plans and three projects had a change in scope. One project had a work stoppage due to this scenario and it took them more than 6 months to respond. Although all these projects had some information available it wasn't enough to quickly and effectively respond to the leadership's questions.

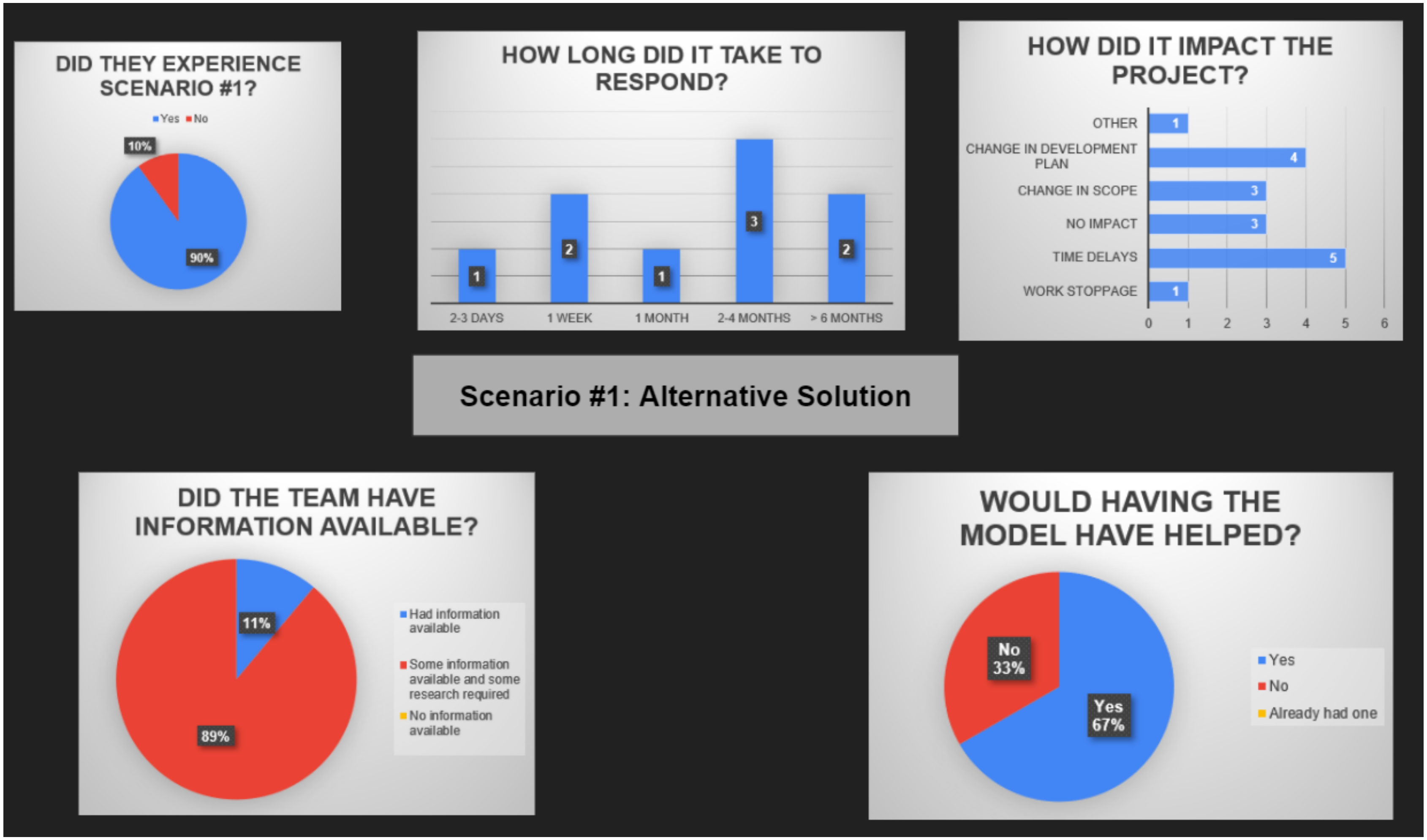


Figure 4.4: Survey Scenario #1 Results

The results of this survey show that this scenario is a common one and can have significant impact on a project. Six of the nine who responded thought a model for evaluating and comparing architectures would have been useful. None of the projects said they had one.

Scenario #2: New Strategic Initiative Results This scenario is less common than the others. Only five respondents said they had experienced this scenario while working on their enterprise data architecture project. Most were large government organizations that experienced this scenario, but one medium sized government organization and one large commercial organization also experienced this scenario.

Based on the responses, when a project encounters this scenario they are likely to take longer than a month to respond. Four of the projects that encountered this scenario had a change in scope and change in development plan. Two of those projects took longer than 6 months to respond. The change in scope and development plan makes sense, since the enterprise data architecture most likely needed to adjust their architecture in order to support the new initiative.

The fifth response is interesting because they had the information they needed. They had already developed objectives with measurement criteria. This project was able to respond within a day to the scenario. This is an interesting data point, because it shows how quickly a response could occur given someone has the information they need. A project using the decision model could have the information they need to respond. The majority of respondents thought that having the projects objectives defined and having criteria for measuring how the project will meet the objectives would have been helpful to have. Using the decision model could save a project months of work.

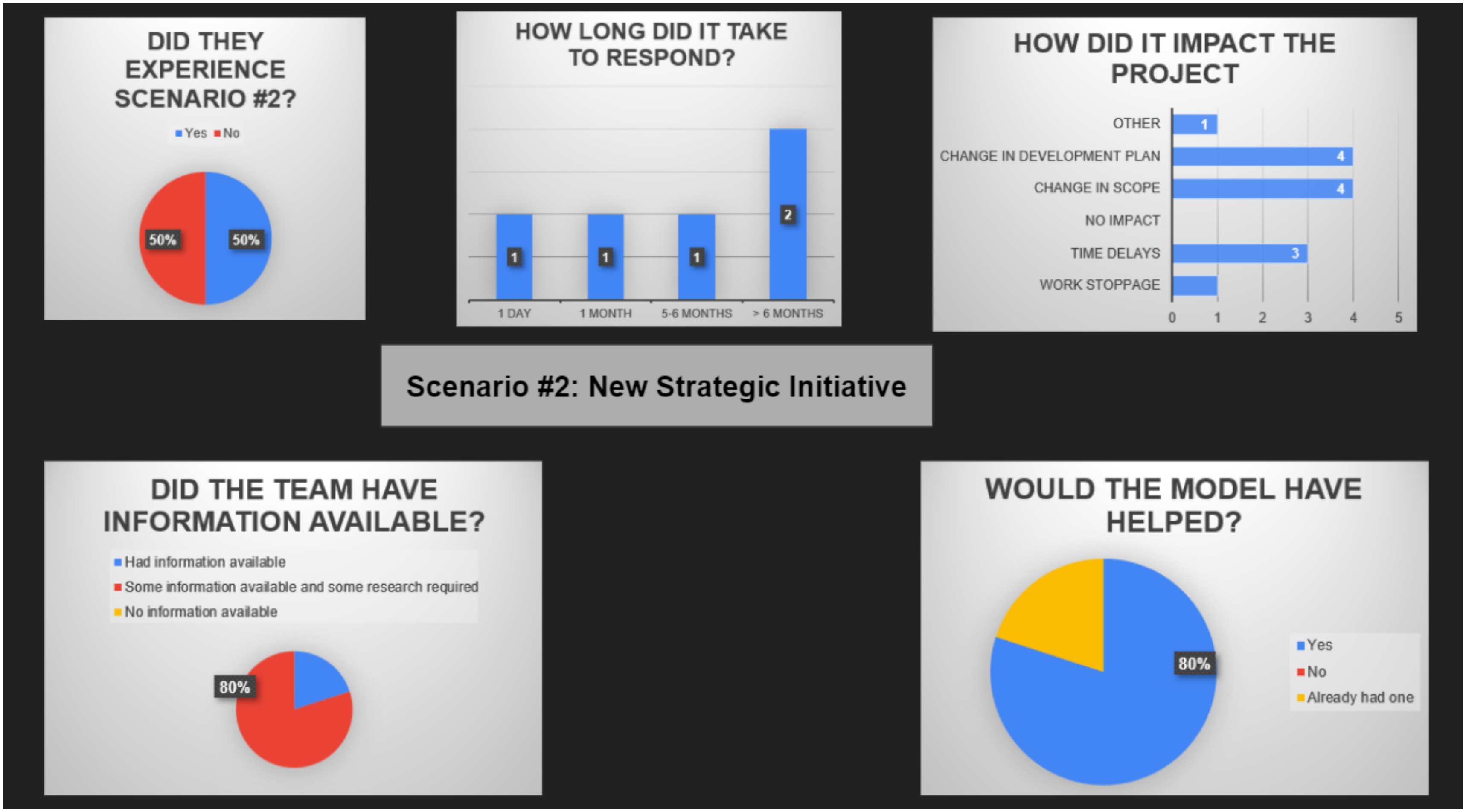


Figure 4.5: Survey Scenario #2 Results

Scenario #3: Scope Increase Results The majority of projects experienced this scenario, nine out of ten selected yes. Of the nine six said they had change in scope, five had time delays, and five had changes in their development plans. There were two projects that had no information when the scenario occurred. One project took a month to respond and the other project took six months to respond. Projects that had some information responded as quickly as 2-3 week and took as long as ¿6 months. There were two projects that had information, one was able to respond in 2-3 days and the other project took a month to respond. Based on these results having data on hand can reduce the response time for this scenario down to a couple days. Compared to when the project only has some information, the quickest response was 2-3 weeks and when there was no information, the quickest response was a month.

One project said they had a value estimate and cost model which could estimate the change in scope's impact on the cost and value. This project still had time delays, change in scope, and change in development plan. This project also took 2-4 months to respond. This data point shows that even with the model, a scope increase scenario can still impact a project. Having a model most likely reduced the impact it could have had on the project, but more data would be needed to draw that conclusion. The majority of respondents thought having a value estimate and cost model would have been helpful for them to have.

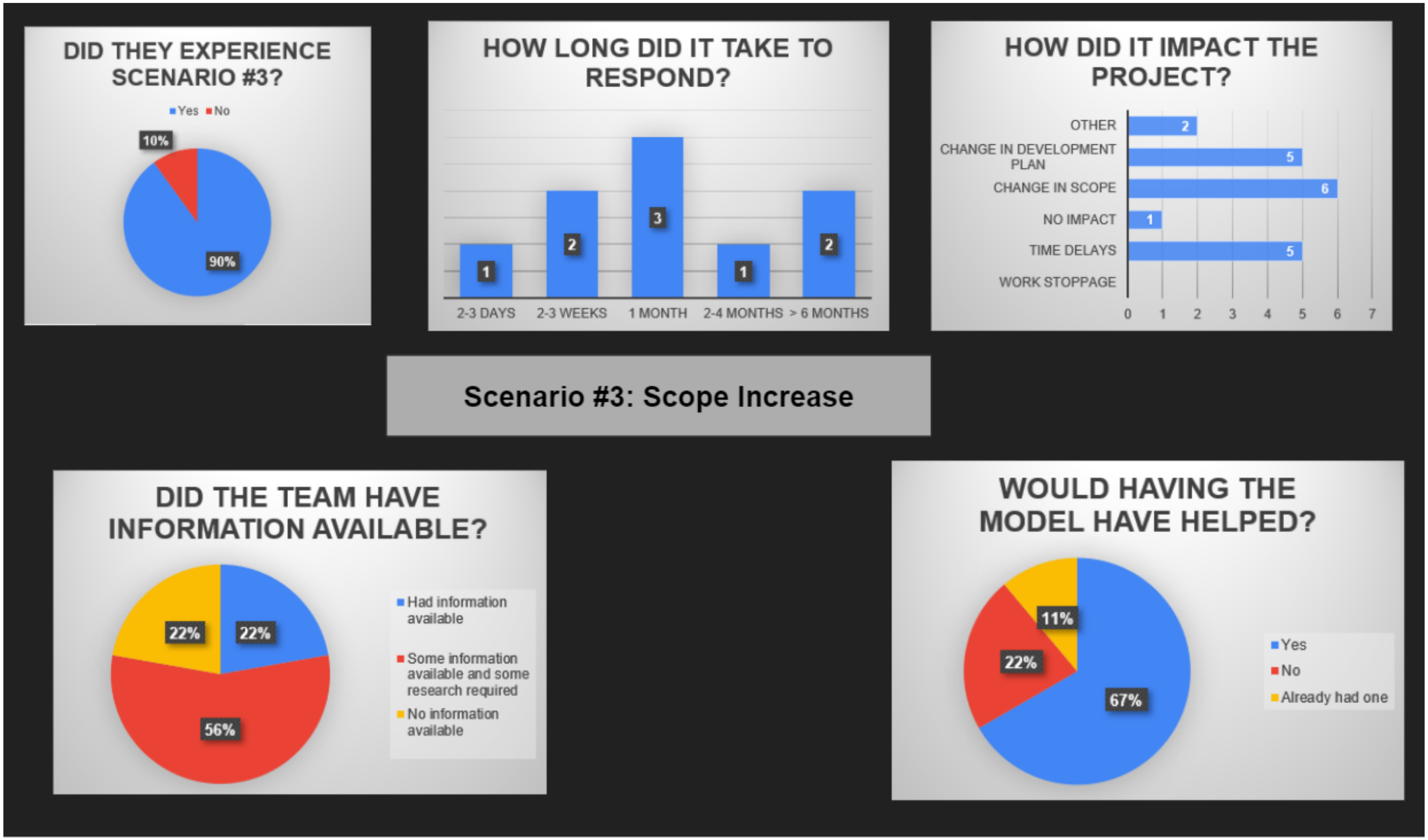


Figure 4.6: Survey Scenario #3 Results

Scenario #4 Results This scenario is common, most projects need to do some cost estimations. The one respondents who didn't respond yes was in the build support role, so they might not have been exposed to the programmatic side of the development project. Most of the people were able to respond to the scenario within 2-3 weeks, which was also the minimum time. One project was not sure how long it took to respond and they selected that they had no information and no impact, so this respondent more than likely wasn't involved in producing the response. The other two responses took significant time to respond. One respondent had no information and took more than six months to respond.

For the three scenarios that had information they were able to respond within the 2-3 week time-frame. For scenarios that had some information, two projects were able to respond in 2-3 weeks. The other two projects that had some information responded in 1 month or 2-4 months. This data shows that the more information a team has, the more quickly they can respond.

The majority of respondents said having a cost model that can estimate development and O&M costs for their project would have been helpful. Seven of the respondents said yes and one respondent already had one. This implies that cost models can be useful in responding to this kind of scenario.

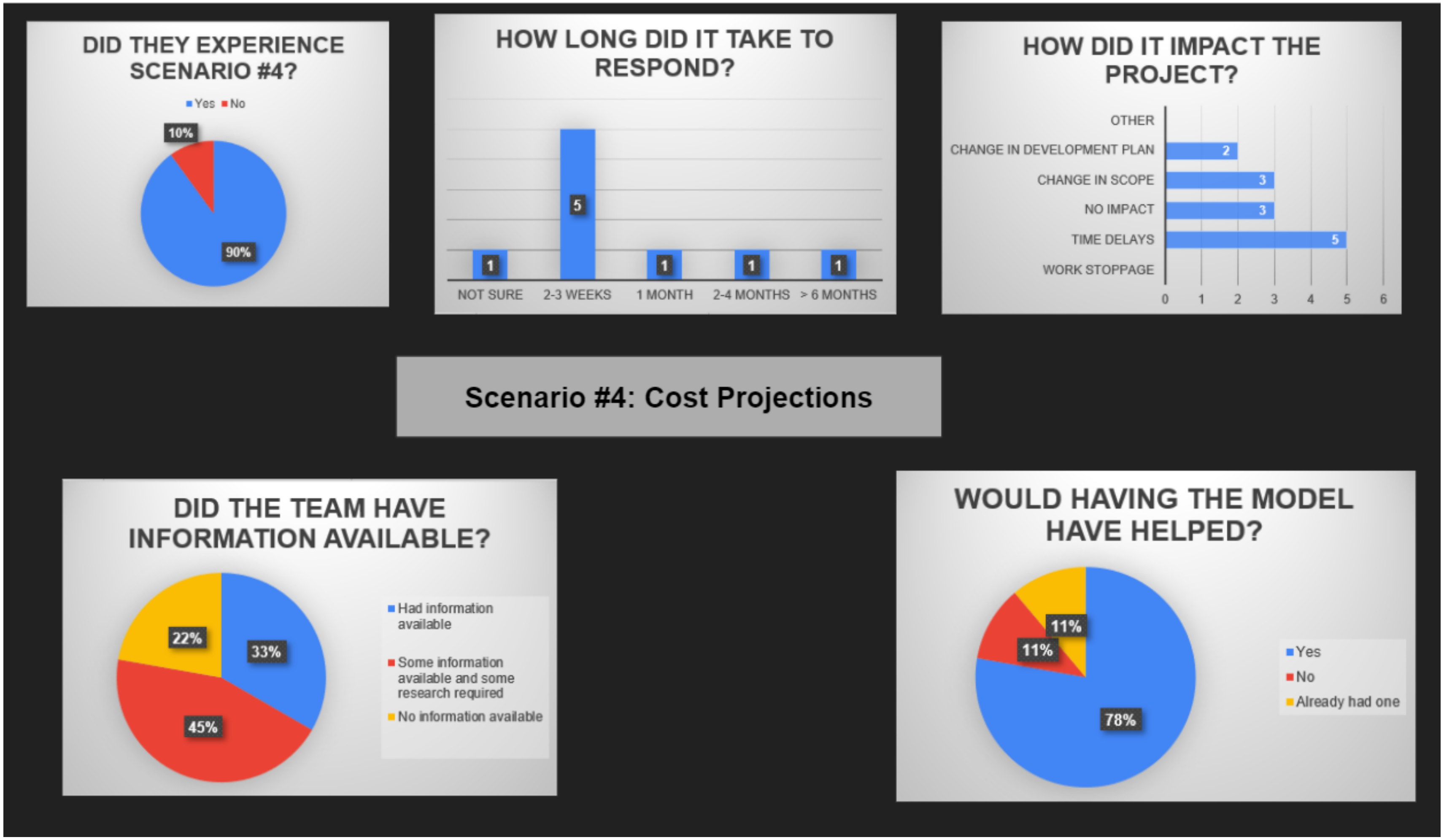


Figure 4.7: Survey Scenario #4 Results

4.4 Evaluation Conclusion

Conclusion Overview This section will first go over how the survey was able to capture the data and information needed in order to address the survey and evaluation goals. Then a discussion of how the results of the survey and evaluation answer the thesis problem statement and hypothesis of the thesis.

4.4.1 Goals Conclusion

Survey Goals Conclusion In section 4.2 the goals for the survey were outlined as key questions. Survey Goal Question Responses:

1. **What development process did they use to develop the enterprise data architecture?**

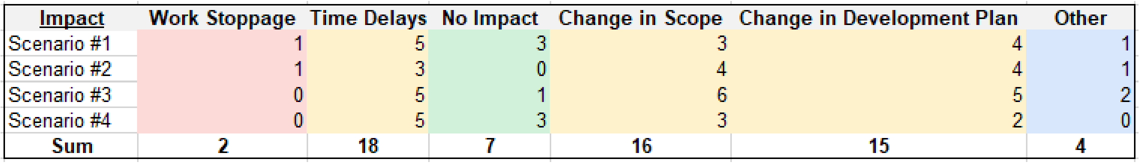
   The majority of projects used an agile development process. A few used V-Model and Waterfall. One used an mix of agile, acquisition, and V-model.

2. **Did development teams encounter these problem scenarios?**

   Yes the majority of development teams did encounter the scenarios. The only scenario that was less commonly encountered was Scenario #2, a New Strategic Initiative.

3. **How was the project impacted by the scenarios?**

Table 4.1: Survey Average Impact

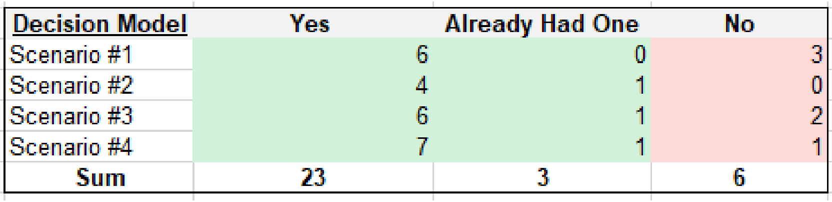


Impact	Work Stoppage	Time Delays	No Impact	Change in Scope	Change in Development Plan	Other
Scenario #1	1	5	3	3	4	1
Scenario #2	1	3	0	4	4	1
Scenario #3	0	5	1	6	5	2
Scenario #4	0	5	3	3	2	0
Sum	**2**	**18**	**7**	**16**	**15**	**4**

The majority of projects had timeline delays, change in scope, and change in development plans. Some projects had work stoppage for a few scenarios. There were some projects that selected “other” which is an unknown impact that the survey didn’t capture. Table 4.1 shows the summery of what impacts projects had from the different scenarios.

4. **Would the decision model have helped reduce the impact, in their opinion?**

Table 4.2: Survey Average Model Response



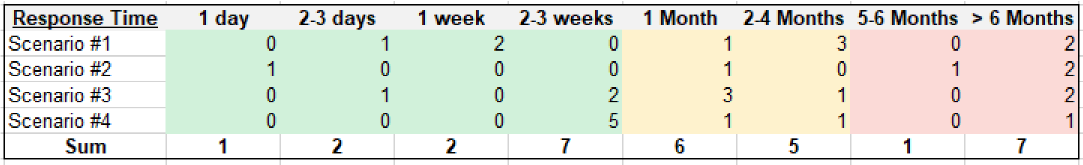
Decision Model	Yes	Already Had One	No
Scenario #1	6	0	3
Scenario #2	4	1	0
Scenario #3	6	1	2
Scenario #4	7	1	1
Sum	23	3	6

The majority of respondents did think that having the different components of the model would have been helpful for them to have when they encountered the different scenarios. Table 4.2 shows the summery of how the people responded across the different scenarios.

Evaluation Goals Conclusion The first goal was the validate that teams developing enterprise data architectures struggle to make decisions, justify their decisions, and respond to their leadership’s questions. Based on the results from the survey, development teams do struggle to make decisions and respond to the scenarios in the survey. Figure 4.3 shows how many iterations an enterprise data architecture can go though and Figure 4.2 shows how long it takes to determine a final architecture design. It can take over a year to determine a final design. But the data also shows that it is possible to design and develop an enterprise data architecture much more quickly.

The next goal was to understand how trouble making decisions, backing up decisions, and responding to questions can impact the development of an enterprise data architecture. Based on the survey response data, the majority of projects struggle to respond to the

Table 4.3: Survey Average Response Time



Response Time	1 day	2-3 days	1 week	2-3 weeks	1 Month	2-4 Months	5-6 Months	> 6 Months
Scenario #1	0	1	2	0	1	3	0	2
Scenario #2	1	0	0	0	1	0	1	2
Scenario #3	0	1	0	2	3	1	0	2
Scenario #4	0	0	0	5	1	1	0	1
Sum	**1**	**2**	**2**	**7**	**6**	**5**	**1**	**7**

different scenarios, where leadership questions need to be answered. Table 4.3 show a summery table of how long on average each team takes to respond to the different scenarios. Green is meant to show a reasonable response time. One to four months is not good, but some questions require research which could take some time. More than five months is a long time to adequately respond to these scenarios. Table 4.1 shows that projects can be impacted in multiple ways.

The last goal was to determine if people who worked on enterprise data architecture project thought the model-based decision framework would have helped them by providing them an analysis framework to help gather data, analyze the data, and respond to their leadership with the information they need. Table 4.2 shows that the majority of people do think having a model would have helped.

4.4.2 Development Processes Results

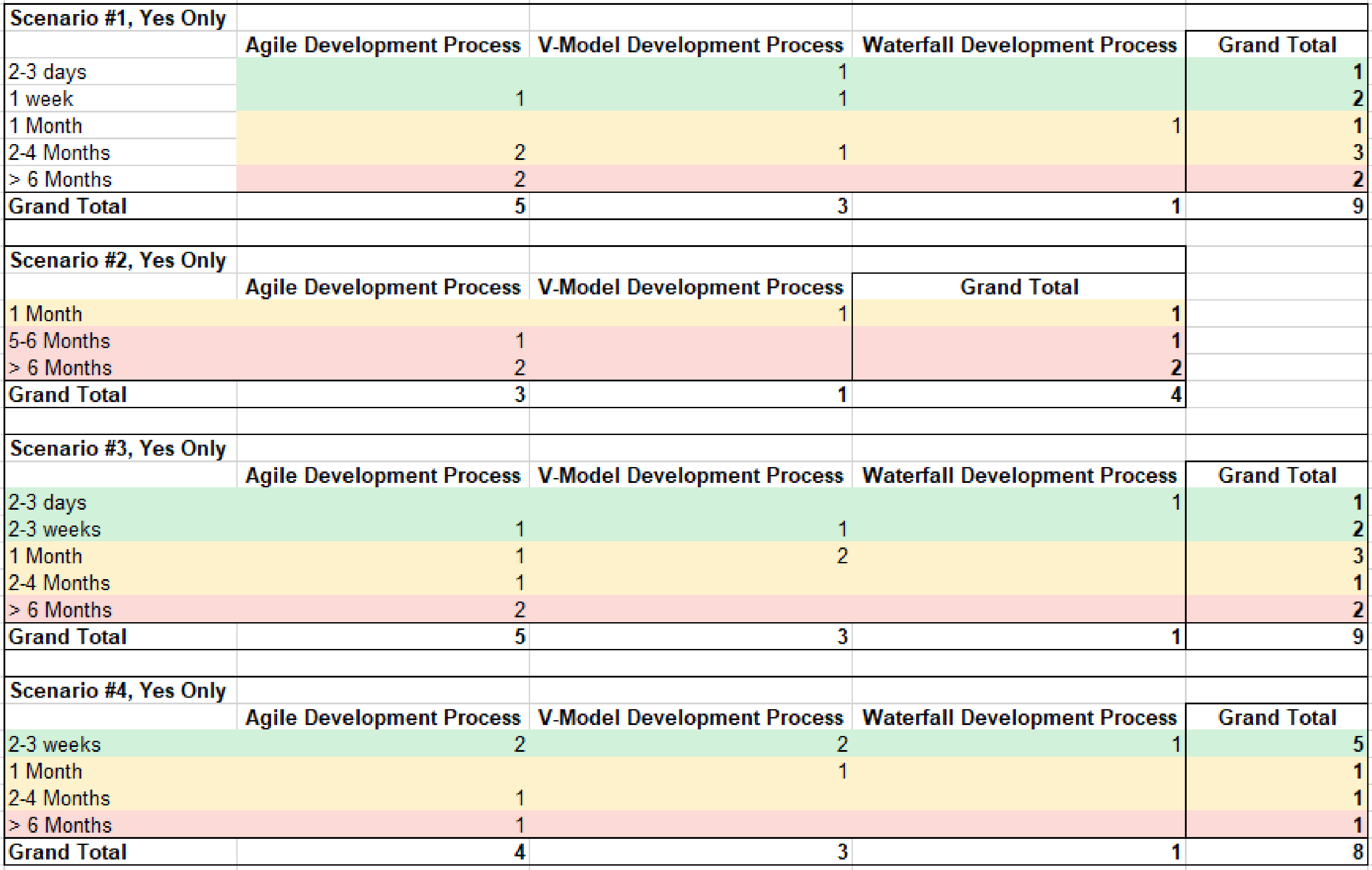
The results from the survey help to better understand how robust the development processes are to the four scenarios. Whether the development process used helped prepare the teams for the different scenarios they encountered and how quickly they were able to respond. Only responses that said yes to each scenario were considered for this analysis.

The survey data needed to be modified slightly analyze the results. One project wasn't considered because they provided a custom development process that didn't fit in the big three development processes. One response which claimed to be a mix of acquisition, v-model, and agile was put into the v-model process category.

For the charts they will be color coded to help visualize and interpret the information.

Green implies a **good** response or outcome, yellow implies a **poor** response or outcome, and red implies a **bad** response or outcome.

Table 4.4: Development Process vs Response Time



Scenario #1, Yes Only				
	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
2-3 days		1		1
1 week	1	1		2
1 Month			1	1
2-4 Months	2	1		3
> 6 Months	2			2
Grand Total	5	3	1	9

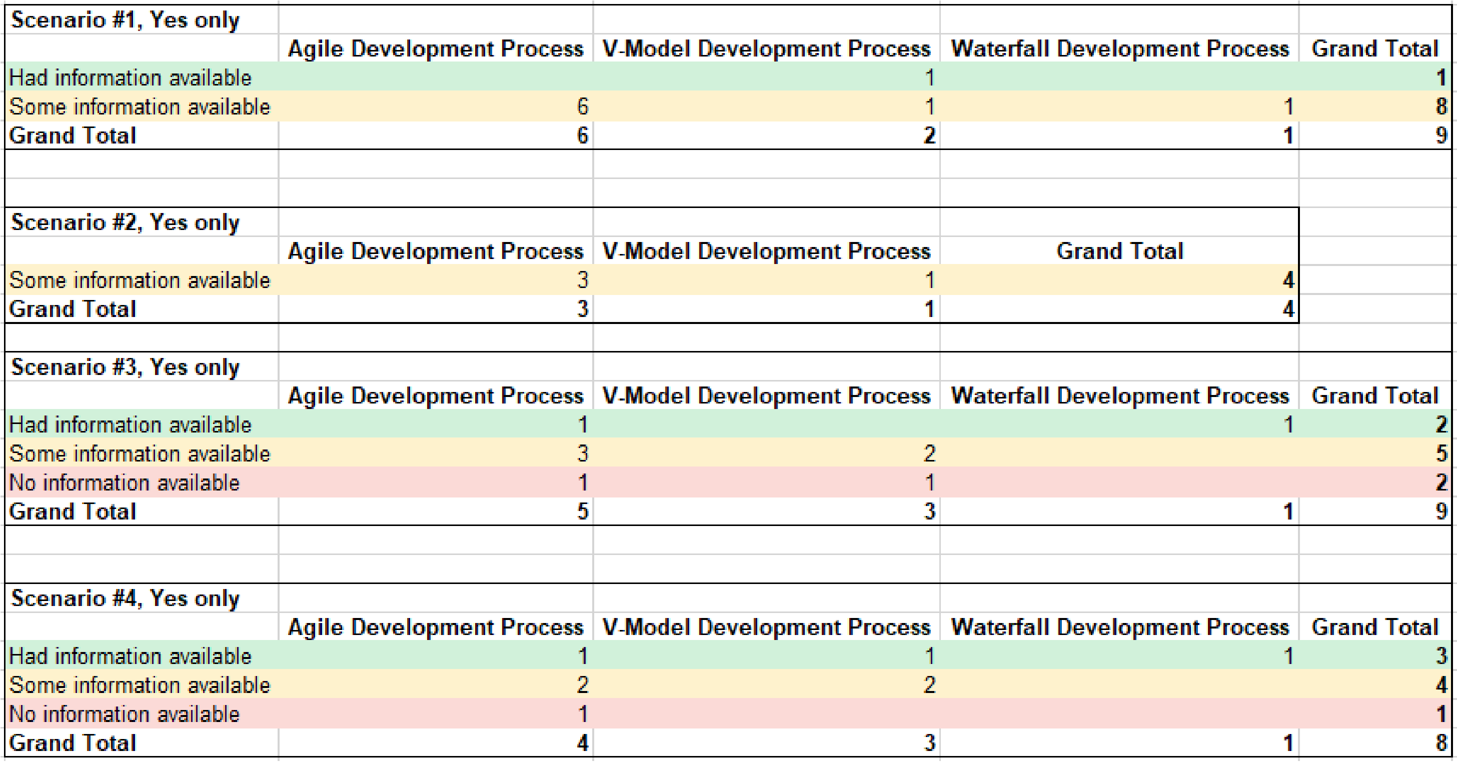
Scenario #2, Yes Only			
	Agile Development Process	V-Model Development Process	Grand Total
1 Month		1	1
5-6 Months	1		1
> 6 Months	2		2
Grand Total	3	1	4

Scenario #3, Yes Only				
	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
2-3 days			1	1
2-3 weeks	1	1		2
1 Month	1	2		3
2-4 Months	1			1
> 6 Months	2			2
Grand Total	5	3	1	9

Scenario #4, Yes Only				
	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
2-3 weeks	2	2	1	5
1 Month		1		1
2-4 Months	1			1
> 6 Months	1			1
Grand Total	4	3	1	8

Development Process Response Times Table 4.4 shows the data on how quickly projects were able to respond to the different scenarios based on the development process they used. For Scenario #1 Agile projects responded poorly while V-Model projects responded better. This is likely because V-Model does some design analysis up front, where agile does not. For Scenario #2 all projects responded poorly, most likely because they do not in their process think about the strategic goals of the organization. The projects using Agile and V-Model didn't respond well to Scenario #3, with the majority in poor and bad response times. All processes responded the best to Scenario #4, but agile still had half of their projects with poor response times. Since the waterfall process only has one data point it's hard to see any potential trends, but similar to the V-Model, there is more design thinking and cost modeling done up front, so that project was able to respond more quickly to most scenarios it encountered.

Table 4.5: Development Process vs Information



Scenario #1, Yes only				
	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
Had information available		1		1
Some information available	6	1	1	8
Grand Total	6	2	1	9

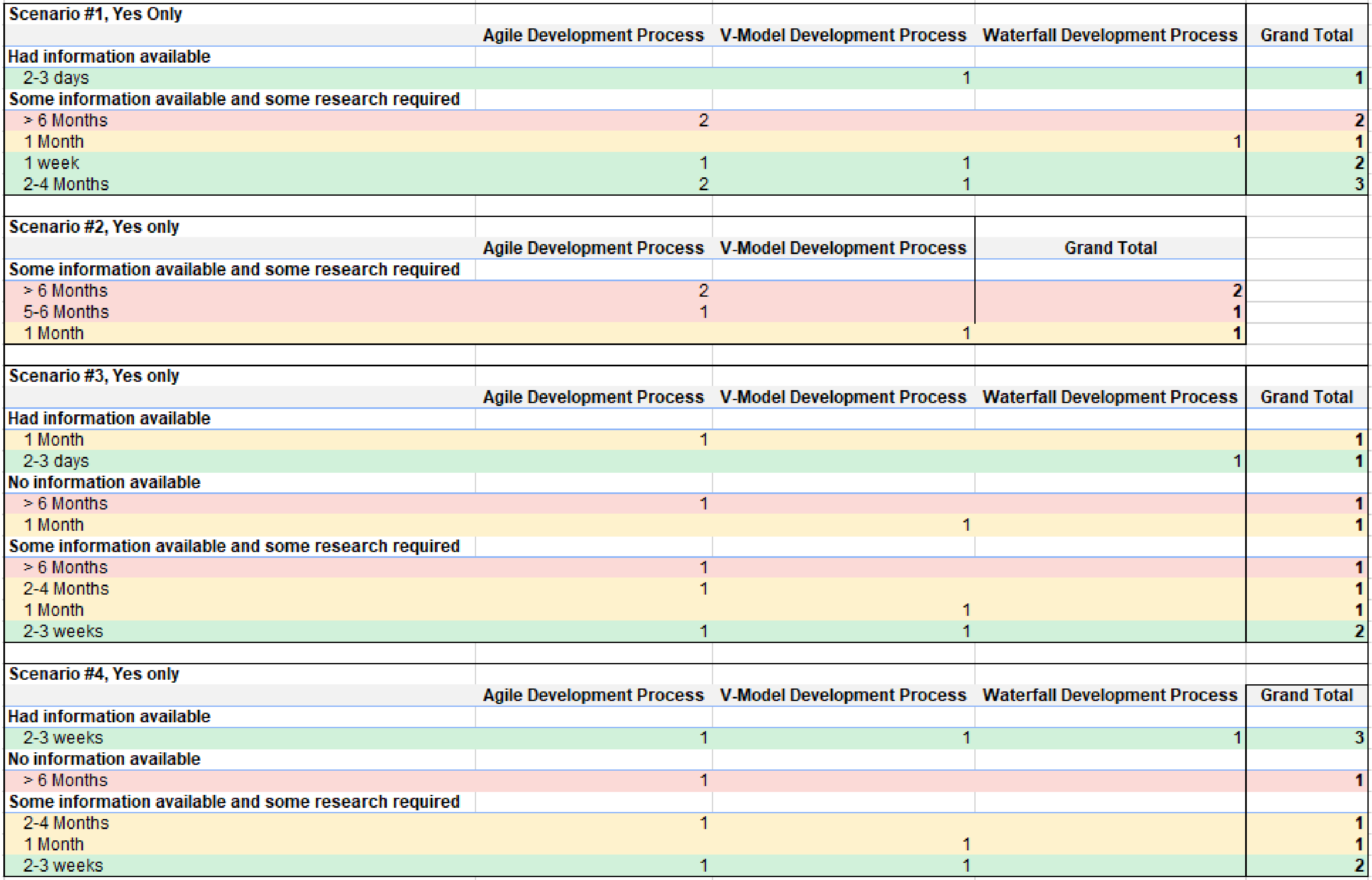
Scenario #2, Yes only			
	Agile Development Process	V-Model Development Process	Grand Total
Some information available	3	1	4
Grand Total	3	1	4

Scenario #3, Yes only				
	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
Had information available	1		1	2
Some information available	3	2		5
No information available	1	1		2
Grand Total	5	3	1	9

Scenario #4, Yes only				
	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
Had information available	1	1	1	3
Some information available	2	2		4
No information available	1			1
Grand Total	4	3	1	8

Development Process Information Table 4.5 shows the data on how much information projects had when they encountered the different scenarios, based on the different development processes. Key take always are most processes provide the teams with some information to handle the four scenarios. Two projects one using Agile and one using V-Model had no information when encountering Scenario #3 and #4. More data is needed to draw any patterns. Waterfall only has one data point, but it had the information on hand for two of the scenarios. The "Some information available and some research required" is a broad option that includes teams that have little information, which still requires a good amount of research and teams that have most information, but still require a little bit of research. Future surveys should include these two more granular options, to help interpret how much information development processes provide teams.

Table 4.6: Development Processes Response Time vs Information



Scenario #1, Yes Only	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
Had information available				
2-3 days		1		1
Some information available and some research required				
> 6 Months	2			2
1 Month			1	1
1 week	1	1		2
2-4 Months	2	1		3

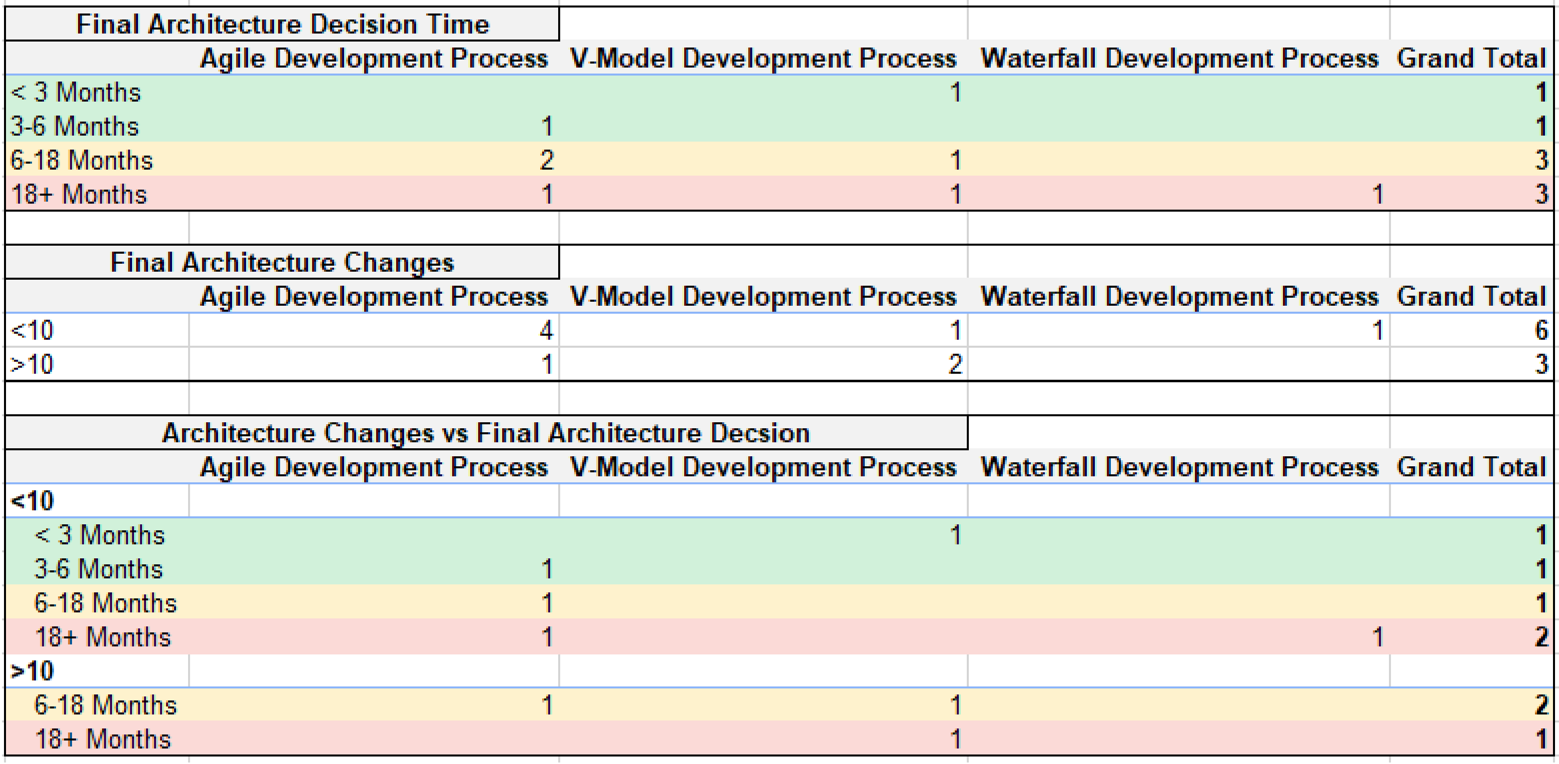
Scenario #2, Yes only	Agile Development Process	V-Model Development Process	Grand Total
Some information available and some research required			
> 6 Months	2		2
5-6 Months	1		1
1 Month		1	1

Scenario #3, Yes only	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
Had information available				
1 Month	1			1
2-3 days			1	1
No information available				
> 6 Months	1			1
1 Month		1		1
Some information available and some research required				
> 6 Months	1			1
2-4 Months	1			1
1 Month		1		1
2-3 weeks	1	1		2

Scenario #4, Yes only	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
Had information available				
2-3 weeks	1	1	1	3
No information available				
> 6 Months	1			1
Some information available and some research required				
2-4 Months	1			1
1 Month		1		1
2-3 weeks	1	1		2

Information Impacts on Response Time Table 4.6 combines the two previous views to look at how having information available effects the response times to the different scenarios. For Scenario's 1, 3, and 4 the projects that had information were able to respond quickly. For projects that had some information and required some research their response times varied considerably. For projects that had no information, their response times were in the poor and bad categories. It does appear that the more information a team has on hand the quicker they can respond and the less research is needed. Future surveys should break up how much information did they have vs how much they needed to research in the options for a response.

Table 4.7: Development Processes Architecture Changes And Decision Time



Final Architecture Decision Time				
	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
< 3 Months		1		**1**
3-6 Months	1			**1**
6-18 Months	2	1		**3**
18+ Months	1	1	1	**3**

Final Architecture Changes				
	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
<10	4	1	1	6
>10	1	2		3

Architecture Changes vs Final Architecture Decsion				
	Agile Development Process	V-Model Development Process	Waterfall Development Process	Grand Total
<10				
< 3 Months		1		**1**
3-6 Months	1			**1**
6-18 Months	1			**1**
18+ Months	1		1	**2**
>10				
6-18 Months	1	1		**2**
18+ Months		1		**1**

Development Process Architecture Decisions Table 4.7 shows both the time it took to determine a final architecture and the number of architecture changes. Most Agile teams took 6-18 months to determine a final architecture with one team taking 3-6 months and another team taking more than 18 months. The V-Model projects hit all the categories: good, poor, and bad. The one project that used waterfall took more than 18 months to finalize their architecture.

The number of architecture changes is difficult to interpret because the survey responses were not consistent. The architecture changes could be minor or major. Future survey's should specify major changes versus minor changes to the architecture.

The last table shows the length of time to determine the final architecture grouped by the number of changes made to the final architecture. Projects that had more then ten changes to their architecture had no projects the finalized their architecture in six months or less. All projects that had more then 10 changes to their architecture took more than six months to finalize the architecture. However, there were still projects that took over 18 months to finalize their architecture even with less than ten the changes. Based on these results, having less then ten changes to the architecture means it is possible to finalize an enterprise data architecture in less than six months. If there are more than 10 changes to the architecture it is more likely that it will take longer than six months to finalize the architecture.

4.4.3 Hypothesis Validation

In section 1.4 the hypothesis stated, "The development process for an enterprise data architecture can be improved by enabling data driven decision making on the architecture design and reducing leadership question response time throughout the development life-cycle."

Based on the results of the survey, teams that make more changes to the final architecture are more likely to have a longer development time. The model-based decision framework helps teams to make data driven decisions on their architecture. Making data driven decisions on the architecture will help reduce the number of changes to the final architecture because any additional requirements or modifications can be analyzed to determine their value and cost before making the change. The decision maker can determine whether the change would be worth it to them.

Reducing the number of changes means the development team is more likely to decide on a final architecture earlier in the development process and complete the project on time. The value model-based decision framework will improve the development teams ability to decide on an architecture and minimize changes, because it is aligned with the values of the organization and is backed by objective information about the architecture.

The results of the survey show that enterprise data architecture development teams do experience leadership questions that can negatively impact the team. The value model-based decision framework can help development teams respond to these scenarios more effectively by using the decision, MBSE, and cost models to capture and analyze information quickly. The response is faster and the information is traceable.

The value model-based decision framework provides development teams a tool to quickly adjust and respond to leadership questions about the project with information backed by data. It gives them a method for consistently assessing the different alternatives, will enable data-driven architecture decisions, and helps the team respond to future alternatives that arise throughout the development life-cycle. The value model-based framework helps teams respond to cost and scope increase questions, because the team has a tool that allows them to make an objective assessment on changes in scope and how that effects schedule and cost. The results from the survey confirm having the information and tool would help improve their ability to respond to the different scenarios that are negatively impacting their project.

Chapter 5: Conclusion

Summary Developing an enterprise data architecture is challenging. Enterprise data architecture projects can take a long time to design and develop to completion. Based on the results of the survey, large organizations are more likely to face challenges making decisions on their enterprise architecture and delays in completing the project.

With current development methods, the chance of a project being delayed due to not having the information needed to respond to leadership questions is 50%. These delays can be reduced if the development teams have the information they need to quickly respond. Based on the results of the survey the respondents thought having components of the value model-based decision framework would have helped them respond to those scenarios.

By combining value-based decision making with model-based systems engineering techniques, the value model-based decision framework can help development teams determine the best enterprise data architecture for their organization. Using the framework will help organize the data collection and analysis of different architectures. It will support future decisions that arise during the development process, which keeps the team focused on delivering an enterprise capability.

5.1 Benefits of a Value Model-Based Decision Framework

The evaluation of the value model-based decision framework in Chapter 4 focused on improving the decision process in two areas. Helping them determine their enterprise data architecture design by using value and data driven decision making to analyze alternative designs. And by enabling development teams with the model and information needed to respond quickly to decision makers questions during the development life-cycle.

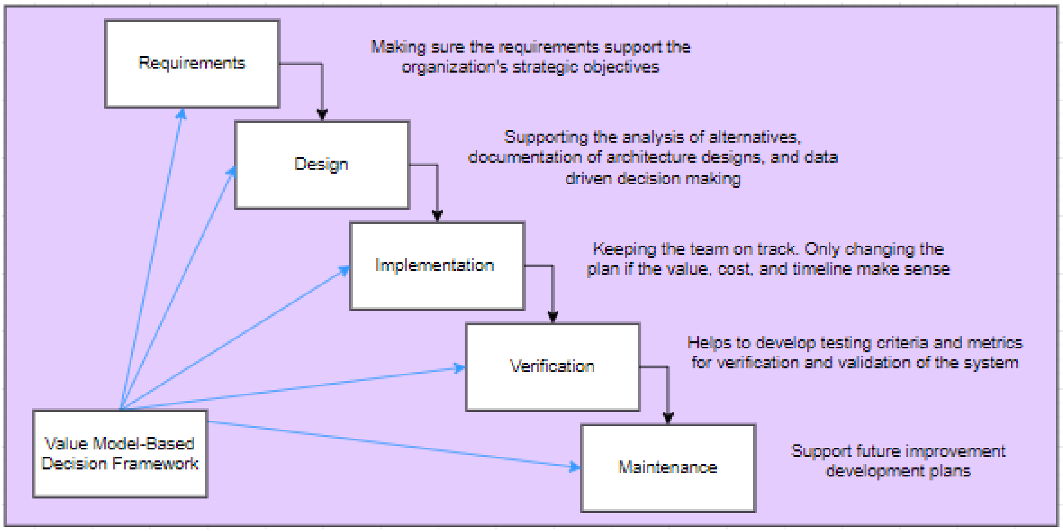

Requirements
Making sure the requirements support the organization's strategic objectives
Design
Supporting the analysis of alternatives, documentation of architecture designs, and data driven decision making
Implementation
Keeping the team on track. Only changing the plan if the value, cost, and timeline make sense
Verification
Helps to develop testing criteria and metrics for verification and validation of the system
Value Model-Based Decision Framework
Maintenance
Support future improvement development plans

Figure 5.1: Value Model and the Development Process

5.1.1 Demonstrated Benefits

Robustness The decision process is able to handle the change in priorities, the change in objectives, or the addition of new solutions. Helps inform the decision maker on what the impact of their decisions are on the enterprise data architecture project. Once the parametric model is built and the decision is made to proceed with an implementation, the model provides a change buffer for the project. If other solutions appear or leadership changes or events occur, the model can help the decision maker understand if they should change their implementation or maintain their current development path. They can quickly compare how alternative implementations are ranked based on the new information. And if there is a new "best value," they can weigh the cost of pivoting against the additional benefit a new solution may provide. This reduces the amount of changes to the project and allows the development team to continue toward the end result. Having the model helps not only make the initial decision on an implementation, it also helps decision makers as the project continues on, by providing that "what-if" analysis capability.

Question Response Speed If a project does not have a decision model, each new question the decision maker has can take a lot of time to research. Each question must be understood, the data collected, and then analyzed. With the decision model, the problem is already contextualized in a tool that can be easily modified. The initial build of the model takes time, but that time is data gathering that needs to occur anyway for a decision maker to make a decision on the implementation and develop the implementation plan. After the model is built many questions the decision maker has can be quickly answered. Additional data gathering and analysis may be required but it is much faster than starting from scratch every time a decision maker has a question. Because it is much faster the decision maker can ask more questions, which leads to the decision maker being better informed and more confident in their decisions.

Both of the benefits are solving the issue of decision maker's not having a tool or process that enables them to have their questions answered or compare alternatives.

Example Decision Maker Questions:

- How does this new alternative compare to the other's we have evaluated already?
- What if we made objective X the highest priority?
- What if we added a new objective, how do the alternative approaches rank? (expanding scope)
- What if we improved X alternative in this area, how would that change the ranking?
- What if we dropped objective X, how would that change the ranking?

Without the value model-based decision framework, questions like this would either not be possible to answer, be answered based on subjective biased information, or take a long time to research each one. The model can be used to help decision makers and the development team stay in sync for what the goals of the project are, which makes it easier for the development team to stay focused on building towards those goals.

5.1.2 Potential Other Benefits

There are likely many other benefits the Value Model-Based Decision Framework could provide teams. Using the framework could help decision makers identify the questions they should be asking. It helps them understand what matters to their organization. It helps to turn subjective information into objective information. It helps decision makers think about what their priorities are and what matters the most to them. Using the framework to help document the priorities and then develop the measurements for the objectives makes it so a decision is based on data not just intuition. It helps decision makers understand what the competing or conflicting objectives are in the decision. It helps develop the essential requirements that must be fulfilled, which helps focus the effort on what's the most important.

It helps to keep the project in scope, because the objectives of what the solution is supposed to be has been documented not just from a functional perspective, but from a organization and decision maker perspective. This means the process is more robust to leadership changes. A new leader can understand how the previous decision maker came to the conclusions they did and will need to offer a new objective, change the priorities of the objectives, or a new alternative solution to evaluate in order to argue that a change in direction should be made for the project. Because the decision has been documented using the framework, it is traceable.

The value results from the model could be used to track the project with the Earned Value Management Method (EVM). As the project continues to build out capability and address requirements for the system, the current value of the system can be assessed. This can then be tracked along with the cost and schedule. This will show management how the system is growing in value as it progresses along.

5.2 Self Critic

The results of the survey were good based on the length of time it was available and the method of exposing it via LinkedIn. It would be better to have more data to evaluate and draw conclusions from. It would be good to see if more commercial industries like health care are encountering the same problems or education institutions.

One impact the survey excluded was cost. The next time a survey is conducted on enterprise data architecture development projects it should include some questions about total cost of developing and maintaining the system. It would be good to know how the different scenarios impact the cost of the projects. Some of the respondents selected "other" as an impact and my best guess is they meant cost.

It would also be good to have some questions that gather data on how decisions are made on these projects. Do they do a lot of research and make a subjective decision or just make quick reaction decisions or do they never make a decision because of continuous scope creep piles on the amount of research that needs to be done. Getting more information on how decision making works for different development teams would have been useful. Models can be biased is brought up as a concern, but decision makers making decisions based on subjective information is also biased. Collecting data on development teams decision making could help compare the different approaches to decision making.

5.3 Future Work Discussion

Using the Model in Practice The next future work would be to use the Value Model-Based Decision Framework on an enterprise data architecture project and see how it would work in practice. It would be good to see what questions decision makers ask once they have a decision model. And see how frequently do they rely on it. It would be interesting to see if it changes the way managers in an organization operate. Want to know how long it takes to build this model for an enterprise data architecture development project and if it improves the development timeline or adds to it. It would be interesting to see what the unexpected benefits are or if it doesn't have the expected impact.

The Value Model-Based Decision Framework aligns the objectives of the system with the goals of the organization. It would be interesting to see what the objectives are for the different projects and how they change depending on an organization and their decision makers. Do organizations have common objectives for their enterprise data architectures? The example used to demonstrate how the model would work, was at a high-level. In practice the model might have more specific objectives that need to be measured. Using the model to understand the design impacts on cost and performance would be very interesting.

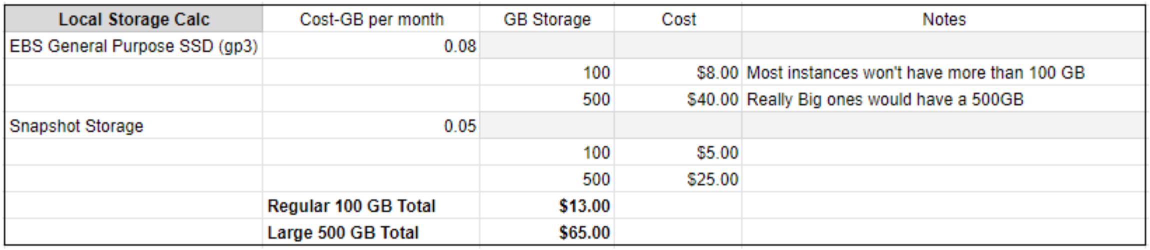
The framework might be useful for focusing in on other decisions an organization will need to make. For example this framework could be applied to deciding what data they should put into the system and how it should be put in. The framework could be used to understand the value a data set provides to an organization. This could be useful for an organization that has a lot of old data that might not be relevant anymore. Data that provides no value to the organization shouldn't be put into the system, which reduces cost.

Modeling Risk Another area of research would be how to model risk. Should risk be modeled as one of the organization's objectives? Or should it be handled outside the value-model. How to capture uncertainty in a decision model could be useful to deciding on which architecture to choose. More research should be done on how to include risk.

Other Software Projects The Value Model-Based Decision Framework could be useful for other complex software development systems. It could be applied to many development projects that have many stakeholders, many options for implementation, and are expensive investments. Future research should be done on expanding this approach to supporting other complex systems.

Appendix A: Appendix

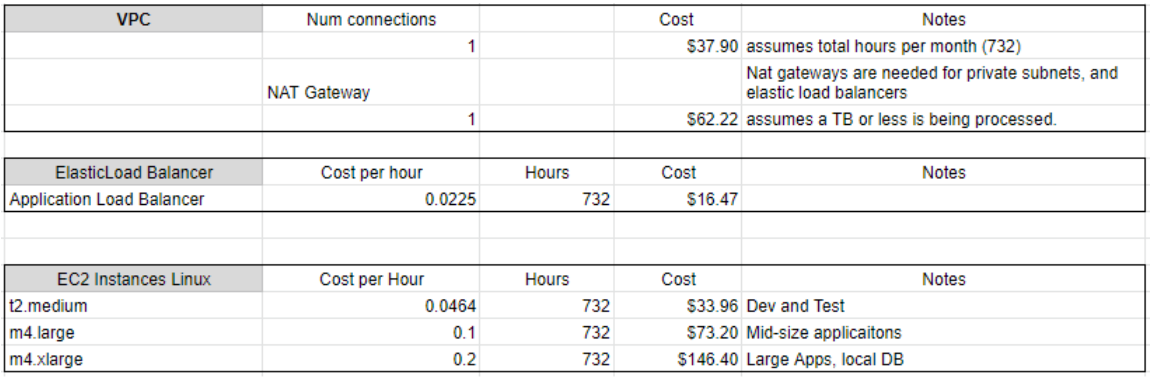
Below are tables describing how costs were calculated for the infrastructure of both the current architecture and the future enterprise data architecture. These cost estimations are simplified versions of what an actual cloud architecture would cost. Architectures are likely to use other services as well, but for the sake of this example the same services were used. All data came from the AWS Ohio region and assumed no free tier was included.



Local Storage Calc	Cost-GB per month	GB Storage	Cost	Notes
EBS General Purpose SSD (gp3)	0.08			
		100	$8.00	Most instances won't have more than 100 GB
		500	$40.00	Really Big ones would have a 500GB
Snapshot Storage	0.05			
		100	$5.00	
		500	$25.00	
	Regular 100 GB Total	**$13.00**		
	Large 500 GB Total	**$65.00**		

Figure A.1: AWS EBS Local Storage Calculations

Table A.1 shows the cost calculation for additional local storage that can be mounted on a cloud instance [71]. The snapshots are backups made of the information stored.



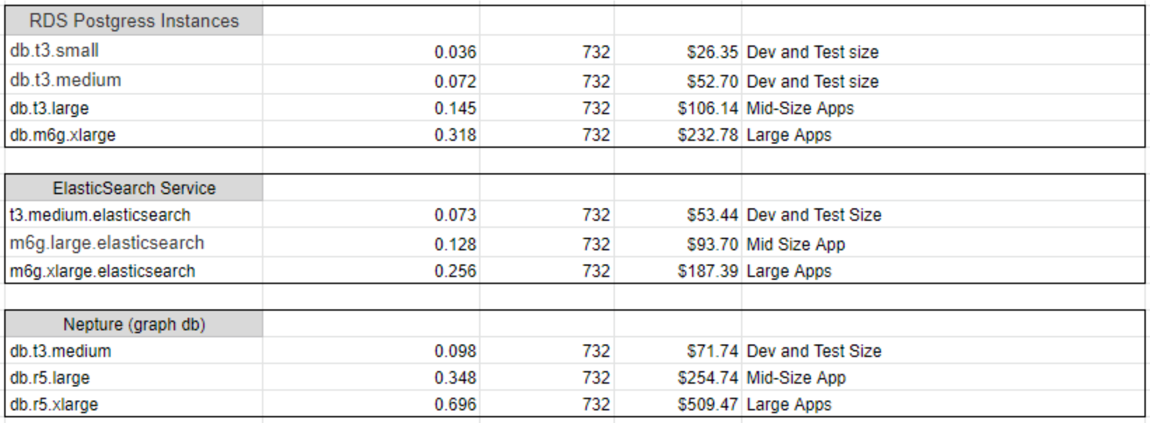
VPC	Num connections		Cost	Notes
	1		$37.90	assumes total hours per month (732)
	NAT Gateway			Nat gateways are needed for private subnets, and elastic load balancers
	1		$62.22	assumes a TB or less is being processed.

ElasticLoad Balancer	Cost per hour	Hours	Cost	Notes
Application Load Balancer	0.0225	732	$16.47	

EC2 Instances Linux	Cost per Hour	Hours	Cost	Notes
t2.medium	0.0464	732	$33.96	Dev and Test
m4.large	0.1	732	$73.20	Mid-size applicaitons
m4.xlarge	0.2	732	$146.40	Large Apps, local DB

Figure A.2: AWS EC2 and VPC Calculations

Table A.2 has the cost estimations for VPC's, ElasticLoad Balancers, and EC2 instances [72] [73] [70]. The EC2 instances are the virtually deployed computers called Elastic Cloud Compute (EC2). All the instances selected are linux based operating systems.

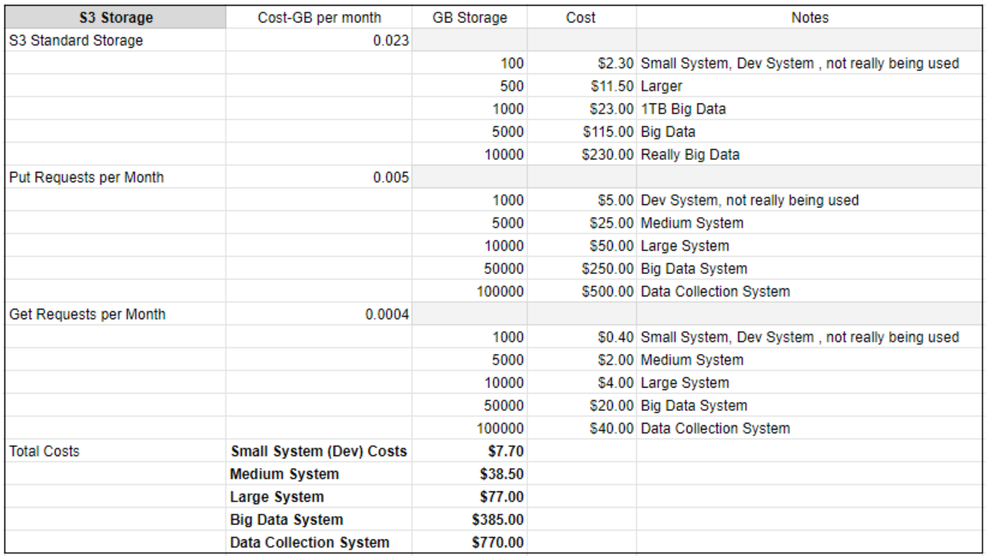


RDS Postgress Instances				
db.t3.small	0.036	732	$26.35	Dev and Test size
db.t3.medium	0.072	732	$52.70	Dev and Test size
db.t3.large	0.145	732	$106.14	Mid-Size Apps
db.m6g.xlarge	0.318	732	$232.78	Large Apps
ElasticSearch Service				
t3.medium.elasticsearch	0.073	732	$53.44	Dev and Test Size
m6g.large.elasticsearch	0.128	732	$93.70	Mid Size App
m6g.xlarge.elasticsearch	0.256	732	$187.39	Large Apps
Nepture (graph db)				
db.t3.medium	0.098	732	$71.74	Dev and Test Size
db.r5.large	0.348	732	$254.74	Mid-Size App
db.r5.xlarge	0.696	732	$509.47	Large Apps

Figure A.3: AWS RDS and other Database Services Calculations

Table A.3 has the costs estimations for relational data base service (RDS) with a PostgreSQL database [74]. It also contains service estimation for ElasticSearch Service which is now called OpenSearch Service [75]. Neptune is the graph database managed service option [76]. All are based on the size of the instance chosen and the amount of time the instance is running.

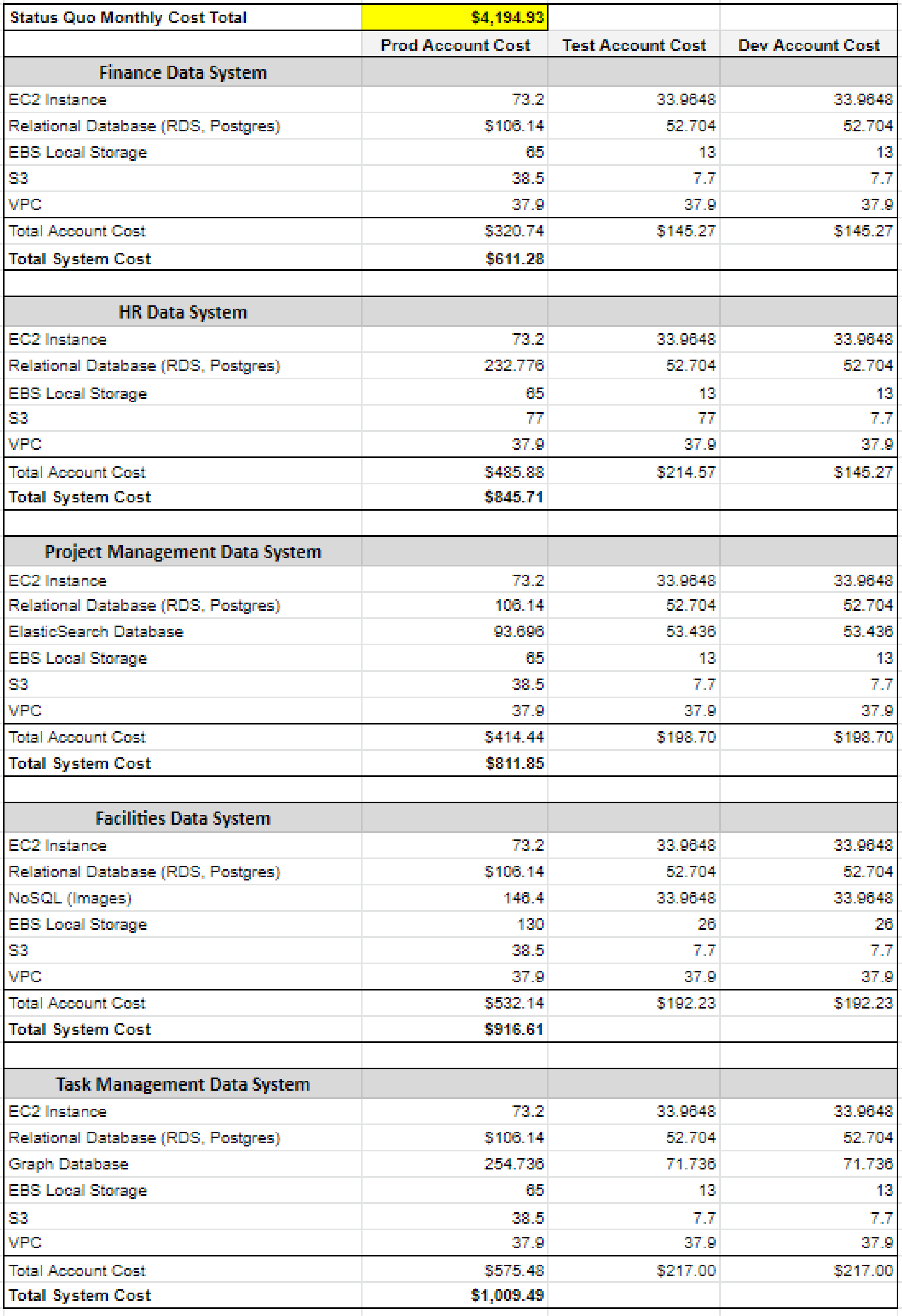
Table A.4 has the cost estimations for data stored in amazon's object storage service called S3 [77]. Costs are estimated based on how much data is stored and how frequently it's accessed.



S3 Storage	Cost-GB per month	GB Storage	Cost	Notes
S3 Standard Storage	0.023			
		100	$2.30	Small System, Dev System , not really being used
		500	$11.50	Larger
		1000	$23.00	1TB Big Data
		5000	$115.00	Big Data
		10000	$230.00	Really Big Data
Put Requests per Month	0.005			
		1000	$5.00	Dev System, not really being used
		5000	$25.00	Medium System
		10000	$50.00	Large System
		50000	$250.00	Big Data System
		100000	$500.00	Data Collection System
Get Requests per Month	0.0004			
		1000	$0.40	Small System, Dev System , not really being used
		5000	$2.00	Medium System
		10000	$4.00	Large System
		50000	$20.00	Big Data System
		100000	$40.00	Data Collection System
Total Costs	**Small System (Dev) Costs**	**$7.70**		
	Medium System	**$38.50**		
	Large System	**$77.00**		
	Big Data System	**$385.00**		
	Data Collection System	**$770.00**		

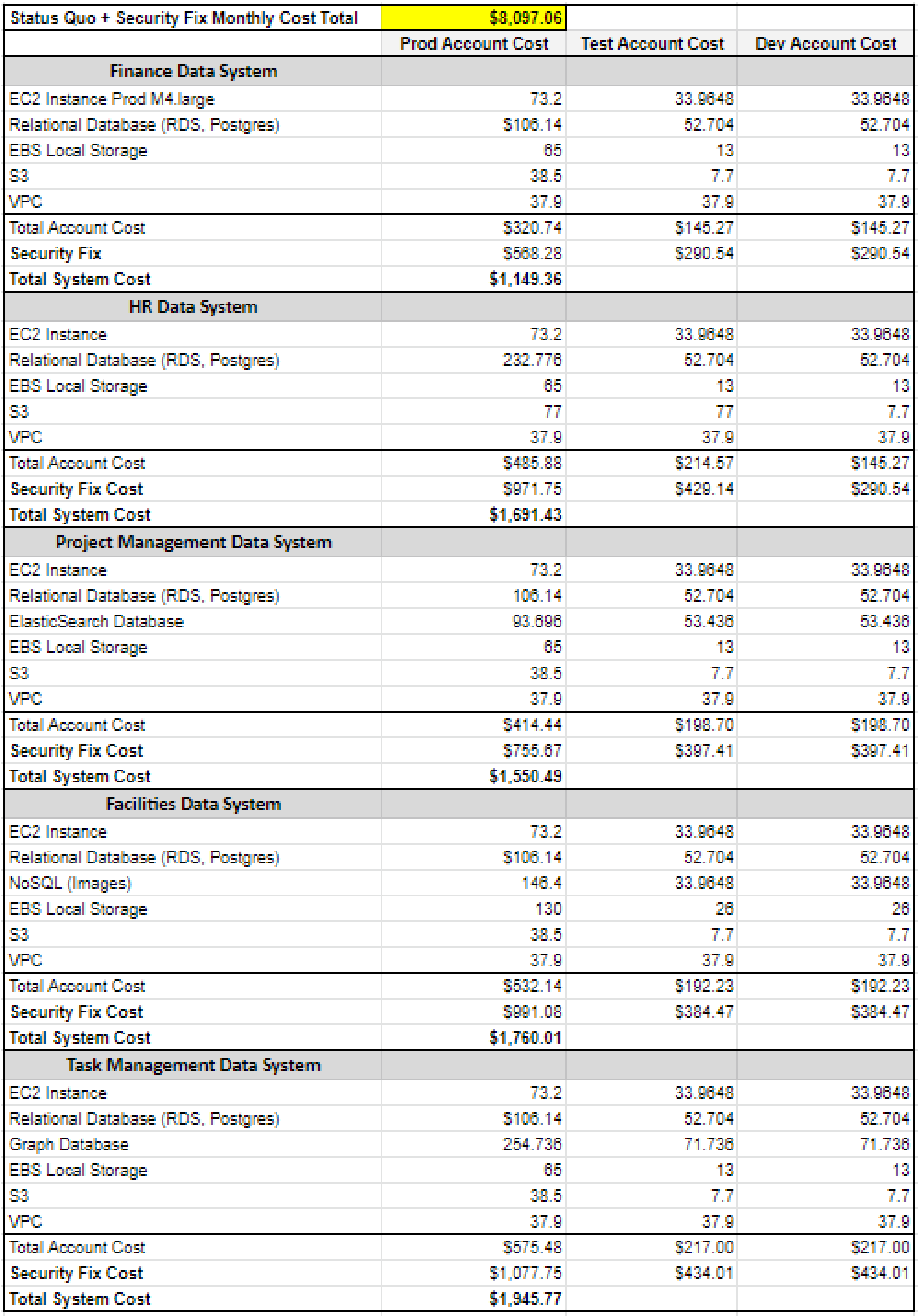
Figure A.4: AWS S3 Storage Calculations

Table A.1: Status Quo Infrastructure Costs



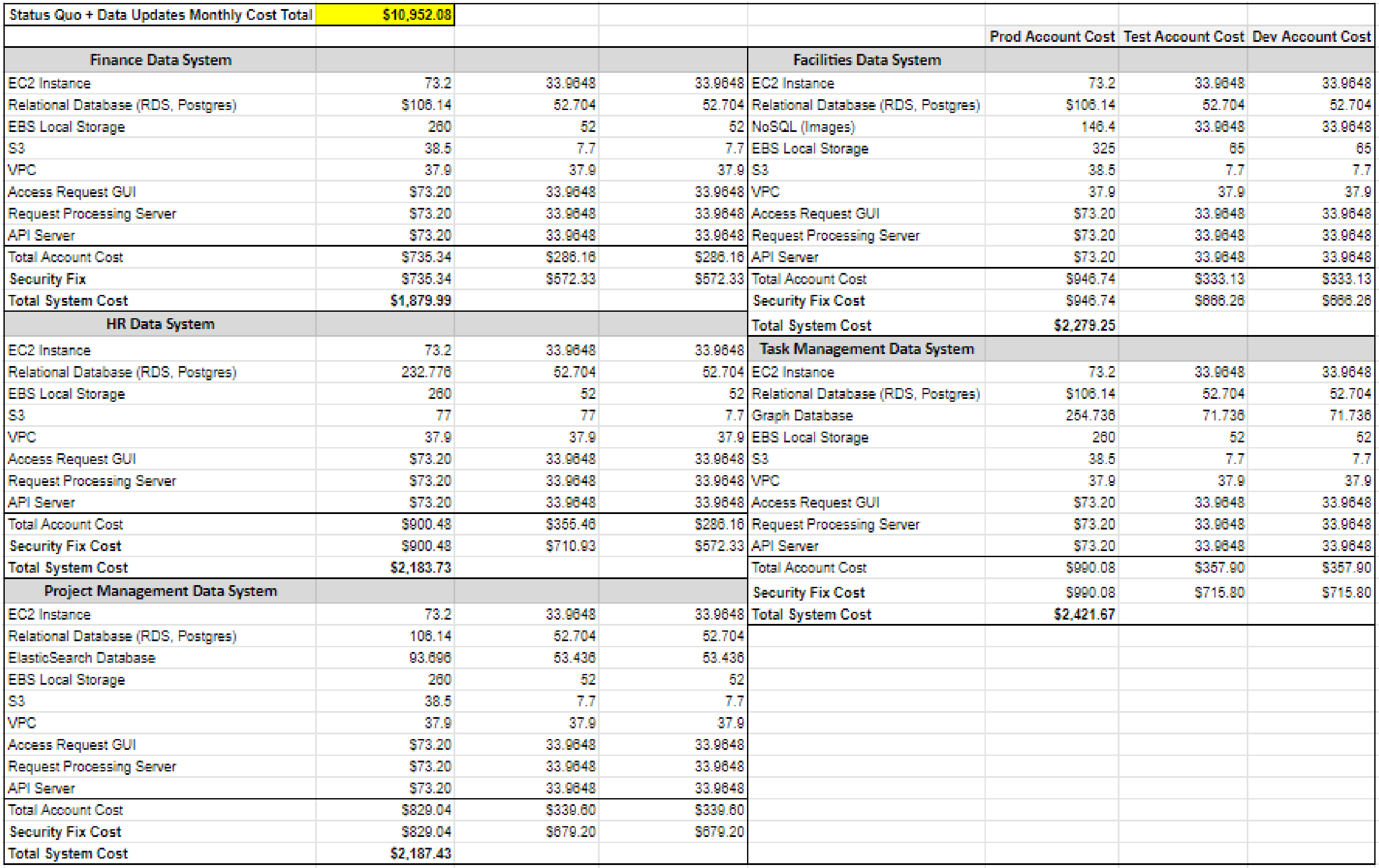
Status Quo Monthly Cost Total	$4,194.93		
	Prod Account Cost	Test Account Cost	Dev Account Cost
Finance Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
EBS Local Storage	65	13	13
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$320.74	$145.27	$145.27
Total System Cost	**$611.28**		
HR Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	232.776	52.704	52.704
EBS Local Storage	65	13	13
S3	77	77	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$485.88	$214.57	$145.27
Total System Cost	**$845.71**		
Project Management Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	106.14	52.704	52.704
ElasticSearch Database	93.696	53.436	53.436
EBS Local Storage	65	13	13
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$414.44	$198.70	$198.70
Total System Cost	**$811.85**		
Facilities Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
NoSQL (Images)	146.4	33.9648	33.9648
EBS Local Storage	130	26	26
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$532.14	$192.23	$192.23
Total System Cost	**$916.61**		
Task Management Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
Graph Database	254.736	71.736	71.736
EBS Local Storage	65	13	13
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$575.48	$217.00	$217.00
Total System Cost	**$1,009.49**		

Table A.2: Status Quo Security Fix Infrastructure Costs



Status Quo + Security Fix Monthly Cost Total	$8,097.06		
	Prod Account Cost	Test Account Cost	Dev Account Cost
Finance Data System			
EC2 Instance Prod M4.large	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
EBS Local Storage	65	13	13
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$320.74	$145.27	$145.27
Security Fix	$568.28	$290.54	$290.54
Total System Cost	**$1,149.36**		
HR Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	232.778	52.704	52.704
EBS Local Storage	65	13	13
S3	77	77	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$485.88	$214.57	$145.27
Security Fix Cost	$971.75	$429.14	$290.54
Total System Cost	**$1,691.43**		
Project Management Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	106.14	52.704	52.704
ElasticSearch Database	93.696	53.436	53.436
EBS Local Storage	65	13	13
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$414.44	$198.70	$198.70
Security Fix Cost	$755.67	$397.41	$397.41
Total System Cost	**$1,550.49**		
Facilities Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
NoSQL (Images)	146.4	33.9648	33.9648
EBS Local Storage	130	26	26
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$532.14	$192.23	$192.23
Security Fix Cost	$991.08	$384.47	$384.47
Total System Cost	**$1,760.01**		
Task Management Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
Graph Database	254.736	71.736	71.736
EBS Local Storage	65	13	13
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$575.48	$217.00	$217.00
Security Fix Cost	$1,077.75	$434.01	$434.01
Total System Cost	**$1,945.77**		

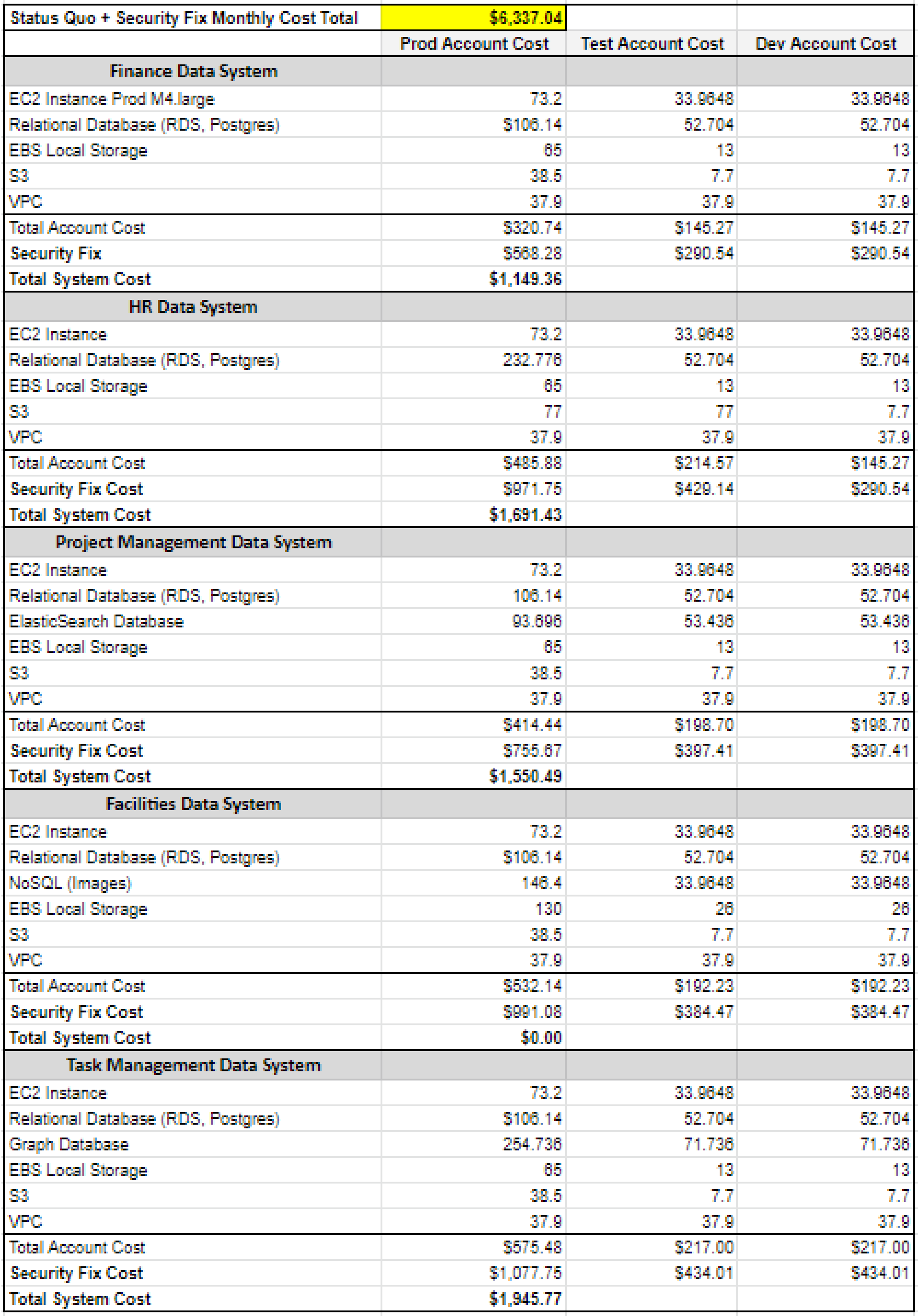
Table A.3: Status Quo Data Updates Infrastructure Costs



Status Quo + Data Updates Monthly Cost Total	$10,952.08		
Finance Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
EBS Local Storage	260	52	52
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Access Request GUI	$73.20	33.9648	33.9648
Request Processing Server	$73.20	33.9648	33.9648
API Server	$73.20	33.9648	33.9648
Total Account Cost	$735.34	$286.16	$286.16
Security Fix	$735.34	$572.33	$572.33
Total System Cost	**$1,879.99**		
HR Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	232.776	52.704	52.704
EBS Local Storage	260	52	52
S3	77	77	7.7
VPC	37.9	37.9	37.9
Access Request GUI	$73.20	33.9648	33.9648
Request Processing Server	$73.20	33.9648	33.9648
API Server	$73.20	33.9648	33.9648
Total Account Cost	$900.48	$355.46	$286.16
Security Fix Cost	$900.48	$710.93	$572.33
Total System Cost	**$2,183.73**		
Project Management Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	106.14	52.704	52.704
ElasticSearch Database	93.696	53.436	53.436
EBS Local Storage	260	52	52
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Access Request GUI	$73.20	33.9648	33.9648
Request Processing Server	$73.20	33.9648	33.9648
API Server	$73.20	33.9648	33.9648
Total Account Cost	$829.04	$339.60	$339.60
Security Fix Cost	$829.04	$679.20	$679.20
Total System Cost	**$2,187.43**		

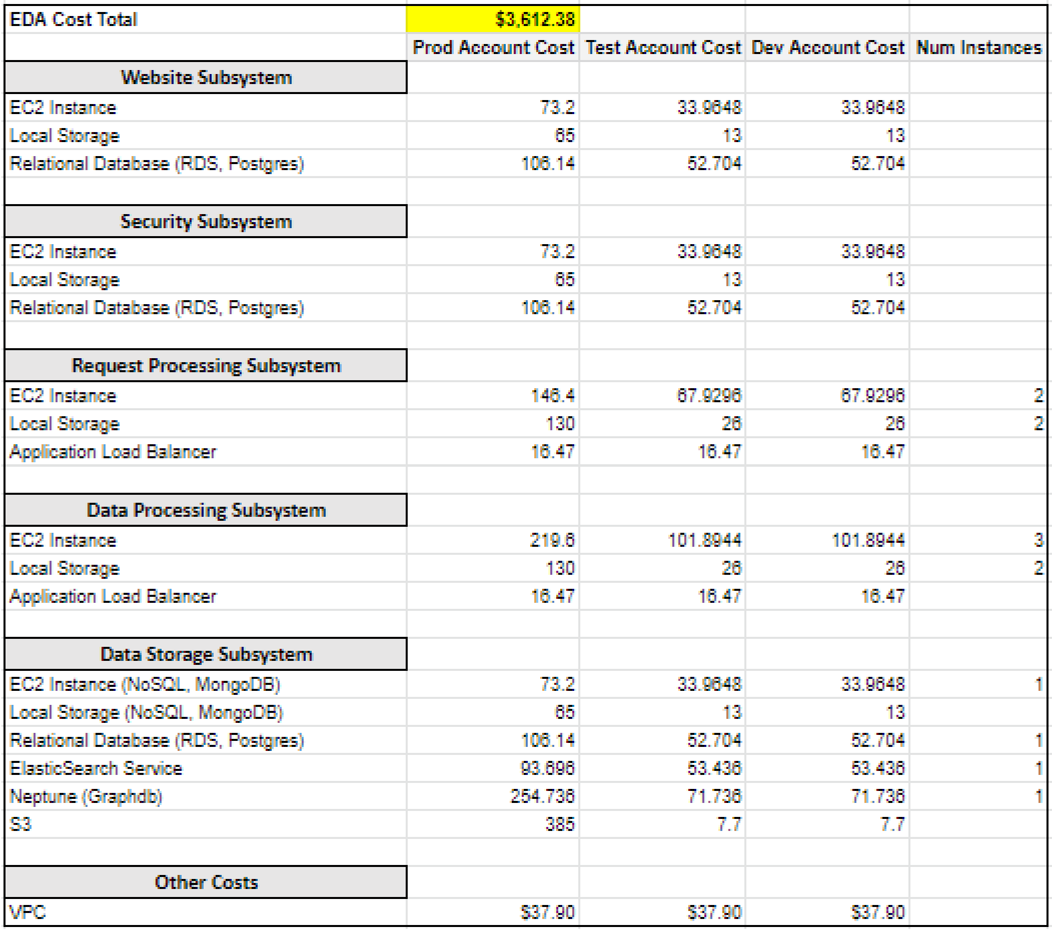
	Prod Account Cost	Test Account Cost	Dev Account Cost
Facilities Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
NoSQL (Images)	146.4	33.9648	33.9648
EBS Local Storage	325	65	65
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Access Request GUI	$73.20	33.9648	33.9648
Request Processing Server	$73.20	33.9648	33.9648
API Server	$73.20	33.9648	33.9648
Total Account Cost	$946.74	$333.13	$333.13
Security Fix Cost	$946.74	$666.26	$666.26
Total System Cost	**$2,279.25**		
Task Management Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
Graph Database	254.736	71.736	71.736
EBS Local Storage	260	52	52
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Access Request GUI	$73.20	33.9648	33.9648
Request Processing Server	$73.20	33.9648	33.9648
API Server	$73.20	33.9648	33.9648
Total Account Cost	$990.08	$357.90	$357.90
Security Fix Cost	$990.08	$715.80	$715.80
Total System Cost	**$2,421.67**		

Table A.4: Status Quo Decommissioned Infrastructure Costs



Status Quo + Security Fix Monthly Cost Total	$6,337.04		
	Prod Account Cost	Test Account Cost	Dev Account Cost
Finance Data System			
EC2 Instance Prod M4.large	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
EBS Local Storage	65	13	13
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$320.74	$145.27	$145.27
Security Fix	$568.28	$290.54	$290.54
Total System Cost	**$1,149.36**		
HR Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	232.778	52.704	52.704
EBS Local Storage	65	13	13
S3	77	77	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$485.88	$214.57	$145.27
Security Fix Cost	$971.75	$429.14	$290.54
Total System Cost	**$1,691.43**		
Project Management Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	106.14	52.704	52.704
ElasticSearch Database	93.696	53.436	53.436
EBS Local Storage	65	13	13
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$414.44	$198.70	$198.70
Security Fix Cost	$755.67	$397.41	$397.41
Total System Cost	**$1,550.49**		
Facilities Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
NoSQL (Images)	146.4	33.9648	33.9648
EBS Local Storage	130	26	26
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$532.14	$192.23	$192.23
Security Fix Cost	$991.08	$384.47	$384.47
Total System Cost	**$0.00**		
Task Management Data System			
EC2 Instance	73.2	33.9648	33.9648
Relational Database (RDS, Postgres)	$106.14	52.704	52.704
Graph Database	254.736	71.736	71.736
EBS Local Storage	65	13	13
S3	38.5	7.7	7.7
VPC	37.9	37.9	37.9
Total Account Cost	$575.48	$217.00	$217.00
Security Fix Cost	$1,077.75	$434.01	$434.01
Total System Cost	**$1,945.77**		

Table A.5: EDA Infrastructure Costs



EDA Cost Total	$3,612.38			
	Prod Account Cost	Test Account Cost	Dev Account Cost	Num Instances
Website Subsystem				
EC2 Instance	73.2	33.9648	33.9648	
Local Storage	65	13	13	
Relational Database (RDS, Postgres)	106.14	52.704	52.704	
Security Subsystem				
EC2 Instance	73.2	33.9648	33.9648	
Local Storage	65	13	13	
Relational Database (RDS, Postgres)	106.14	52.704	52.704	
Request Processing Subsystem				
EC2 Instance	146.4	67.9296	67.9296	2
Local Storage	130	26	26	2
Application Load Balancer	16.47	16.47	16.47	
Data Processing Subsystem				
EC2 Instance	219.6	101.8944	101.8944	3
Local Storage	130	26	26	2
Application Load Balancer	16.47	16.47	16.47	
Data Storage Subsystem				
EC2 Instance (NoSQL, MongoDB)	73.2	33.9648	33.9648	1
Local Storage (NoSQL, MongoDB)	65	13	13	
Relational Database (RDS, Postgres)	106.14	52.704	52.704	1
ElasticSearch Service	93.696	53.436	53.436	1
Neptune (Graphdb)	254.736	71.736	71.736	1
S3	385	7.7	7.7	
Other Costs				
VPC	$37.90	$37.90	$37.90	

Bibliography

Bibliography

[1] S. Theodoridis, "Chapter 3 - Learning in Parametric Modeling: Basic Concepts and Directions," in *Machine Learning*, S. Theodoridis, Ed. Oxford: Academic Press, Jan. 2015, pp. 53–103. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128015223000033

[2] S. Filipe, C. Santos, and M. Pinheiro, *40 Liberec proceedings LEF 2021*, Sep. 2021.

[3] NIST Big Data Public Working Group Use Cases and Requirements Subgroup, "NIST Big Data Interoperability Framework: Volume 3, Use Cases and General Requirements," National Institute of Standards and Technology, Tech. Rep. NIST SP 1500-3, Oct. 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-3.pdf

[4] J. Koo, t. l. w. o. i. a. n. w. Link to external site, and G. Kang, "Security and Privacy in Big Data Life Cycle: A Survey and Open Challenges," *Sustainability*, vol. 12, no. 24, p. 10571, 2020, num Pages: 10571 Place: Basel, Switzerland Publisher: MDPI AG. [Online]. Available: http://www.proquest.com/docview/2471625906/abstract/607A3A9B8AB54682PQ/1

[5] M. Razavian, B. Paech, and A. Tang, "Empirical research for software architecture decision making: An analysis," *Journal of Systems and Software*, vol. 149, pp. 360–381, Mar. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S016412121830267X

[6] "Exploring the Value of MBSE at NASA | APPEL Knowledge Services." [Online]. Available: https://appel.nasa.gov/2018/06/08/exploring-the-value-of-mbse-at-nasa/

[7] "Systems Engineering Definition." [Online]. Available: https://www.incose.org/about-systems-engineering/system-and-se-definition/systems-engineering-definition

[8] N. Hajli, F. Shirazi, M. Tajvidi, and N. Huda, "Towards an Understanding of Privacy Management Architecture in Big Data: An Experimental Research," *British Journal of Management*, vol. 32, no. 2, pp. 548–565, 2021, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8551.12427. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8551.12427

[9] K. Henderson and A. Salado, "Value and benefits of model-based systems engineering (MBSE): Evidence from the literature," *Systems Engineering*, vol. 24, no. 1, pp. 51–66, 2021, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sys.21566. [Online]. Available: http://onlinelibrary.wiley.com/doi/abs/10.1002/sys.21566

[10] K. M. Anderson, “Embrace the Challenges: Software Engineering in a Big Data World,” in *2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering*, May 2015, pp. 19–25.

[11] A. Kerr, “How Do We Manage the Data of the Future?” in *2005 IEEE International Symposium on Mass Storage Systems and Technology*, Jun. 2005, pp. 1–2.

[12] NIST Big Data Public Working Group Definitions and Taxonomies Subgroup, “NIST Big Data Interoperability Framework: Volume 1, Definitions,” National Institute of Standards and Technology, Tech. Rep. NIST SP 1500-1, Oct. 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-1.pdf

[13] “Solution Path for Modernizing Analytic Architectures.” [Online]. Available: https://www.gartner.com/en

[14] M. Gupta and J. F. George, “Toward the development of a big data analytics capability,” *Information & Management*, vol. 53, no. 8, pp. 1049–1064, Dec. 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0378720616300787

[15] NIST Big Data Public Working Group Security and Privacy Subgroup, “NIST Big Data Interoperability Framework: Volume 4, Security and Privacy,” National Institute of Standards and Technology, Tech. Rep. NIST SP 1500-4, Oct. 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-4.pdf

[16] S. Riaz, A. H. Khan, M. Haroon, S. Latif, and S. Bhatti, “Big Data Security and Privacy: Current Challenges and Future Research perspective in Cloud Environment,” in *2020 International Conference on Information Management and Technology (ICIMTech)*, Aug. 2020, pp. 977–982.

[17] “Create a Data Strategy to Ensure Success in Machine Learning Initiatives.” [Online]. Available: https://www.gartner.com/en

[18] U. Narayanan, V. Paul, and S. Joseph, “A novel system architecture for secure authentication and data sharing in cloud enabled Big Data Environment,” *Journal of King Saud University - Computer and Information Sciences*, May 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157820303700

[19] “Solution Path for Building a Holistic Data Management and Analytics Architecture.” [Online]. Available: https://www.gartner.com/en

[20] “Implementing the Technical Architecture for Master Data Management.” [Online]. Available: https://www.gartner.com/en

[21] S. Shah, C. B. Soriano, and A. D. Coutroubis, “Is big data for everyone? the challenges of big data adoption in SMEs,” in *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Dec. 2017, pp. 803–807, iSSN: 2157-362X.

[22] N. Sheikh, “Big Data, Hadoop, and Cloud Computing,” in *Implementing Analytics*. Elsevier, 2013, pp. 185–197. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780124016965000116

[23] NIST Big Data Public Working Group Reference Architecture Subgroup, "NIST Big Data Interoperability Framework: Volume 6, Reference Architecture," National Institute of Standards and Technology, Tech. Rep. NIST SP 1500-6, Oct. 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-6.pdf

[24] F. Daneshgar, G. C. Low, and L. Worasinchai, "An investigation of 'build vs. buy' decision for software acquisition by small to medium enterprises," *Information and Software Technology*, vol. 55, no. 10, pp. 1741–1750, Oct. 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0950584913000839

[25] C. Avci, B. Tekinerdogan, and I. N. Athanasiadis, "Software architectures for big data: a systematic literature review," *Big Data Analytics*, vol. 5, no. 1, p. 5, Aug. 2020. [Online]. Available: https://doi.org/10.1186/s41044-020-00045-1

[26] J. Al-Jaroodi, B. Hollein, and N. Mohamed, "Applying software engineering processes for big data analytics applications development," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan. 2017, pp. 1–7.

[27] A. Tang, J. Han, and R. Vasa, "Software Architecture Design Reasoning: A Case for Improved Methodology Support," *IEEE Software*, vol. 26, no. 2, pp. 43–49, Apr. 2009, num Pages: 43-49 Place: Los Alamitos, United States Publisher: IEEE Computer Society. [Online]. Available: https://www.proquest.com/docview/215837417/abstract/9BBF960F47E245DBPQ/1

[28] I. Groher and R. Weinreich, "A Study on Architectural Decision-Making in Context," in *2015 12th Working IEEE/IFIP Conference on Software Architecture*, May 2015, pp. 11–20.

[29] M. L. Despa, "Comparative study on software development methodologies," no. 3, p. 21, 2014.

[30] S. Soobia.et.al., "Analysis of Software Development Methodologies," *International Journal of Computing and Digital Systems*, vol. 8, no. 5, pp. 445–460, Jan. 2019. [Online]. Available: https://journal.uob.edu.bh/handle/123456789/3583

[31] S. Wambler, "2018 IT Project Success Rates Survey Results," 2018. [Online]. Available: http://www.ambysoft.com/surveys/success2018.html

[32] "Principles behind the Agile Manifesto," 2001. [Online]. Available: https://agilemanifesto.org/principles.html

[33] R. Laigner, M. Kalinowski, S. Lifschitz, R. Salvador Monteiro, and D. de Oliveira, "A Systematic Mapping of Software Engineering Approaches to Develop Big Data Systems," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug. 2018, pp. 446–453.

[34] B. Shahzad, A. M. Abdullatif, N. Ikram, and A. Mashkoor, "Build Software or Buy: A Study on Developing Large Scale Software," *IEEE Access*, vol. 5, pp. 24 262–24 274, 2017, conference Name: IEEE Access.

[35] L. R. Vijayasarathy and C. W. Butler, “Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter?” *IEEE Software*, vol. 33, no. 5, pp. 86–94, Sep. 2016, conference Name: IEEE Software.

[36] D. W. W. Rovce, “MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS,” p. 11, 1970.

[37] J. S. Van Der Ven and J. Bosch, “Busting Software Architecture Beliefs: A Survey on Success Factors in Architecture Decision Making,” in *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug. 2016, pp. 42–49, iSSN: 2376-9505.

[38] “Manifesto for Agile Software Development,” 2001. [Online]. Available: https://agilemanifesto.org/

[39] H.-M. Chen, R. Kazman, S. Haziyev, and O. Hrytsay, “Big Data System Development: An Embedded Case Study with a Global Outsourcing Firm,” in *2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering*, May 2015, pp. 44–50.

[40] O. Gomez, P. Wriedt, and F. Zhao, “Build or Buy: A Case Study for ERP System Selection in SMEs,” in *Human-Computer Interaction. Theory, Design, Development and Practice*, ser. Lecture Notes in Computer Science, M. Kurosu, Ed. Cham: Springer International Publishing, 2016, pp. 23–33.

[41] H.-M. Chen, R. Kazman, and S. Haziyev, “Agile Big Data Analytics Development: An Architecture-Centric Approach,” in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, Jan. 2016, pp. 5378–5387, iSSN: 1530-1605.

[42] R. Wojcik, F. Bachmann, L. Bass, P. Clements, P. Merson, R. Nord, and B. Wood, “Attribute-Driven Design (ADD), Version 2.0:,” Defense Technical Information Center, Fort Belvoir, VA, Tech. Rep., Nov. 2006. [Online]. Available: http://www.dtic.mil/docs/citations/ADA460414

[43] H.-M. Chen, R. Kazman, and S. Haziyev, “Strategic Prototyping for Developing Big Data Systems,” *IEEE Software*, vol. 33, no. 2, pp. 36–43, Mar. 2016, conference Name: IEEE Software.

[44] H. Cervantes, U. A. Metropolitana–Iztapalapa, and R. Kazman, “ADD 3.0: Rethinking Drivers and Decisions in the Design Process,” p. 77.

[45] S. Bellomo, I. Gorton, and R. Kazman, “Toward Agile Architecture: Insights from 15 Years of ATAM Data,” *IEEE Software*, vol. 32, no. 5, pp. 38–45, Sep. 2015, conference Name: IEEE Software.

[46] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, “The Architecture Tradeoff Analysis Method,” p. 11.

[47] A. Tang, M. Razavian, B. Paech, and T.-M. Hesse, “Human Aspects in Software Architecture Decision Making: A Literature Review,” in *2017 IEEE International Conference on Software Architecture (ICSA)*, Apr. 2017, pp. 107–116.

[48] H. van Vliet and A. Tang, “Decision making in software architecture,” *Journal of Systems and Software*, vol. 117, pp. 638–644, Jul. 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0164121216000157

[49] E. Breen, “Thinking Fast and Slow By Daniel Kahneman. Penguin. 2012. £10.99 (pb). 512 pp. ISBN 9780141033570,” *The British Journal of Psychiatry*, vol. 213, pp. 563–564, Sep. 2018.

[50] G. Klein, *Streetlights and Shadows: Searching for the Keys to Adaptive Decision Making*. Cambridge, MA, USA: A Bradford Book, Sep. 2009.

[51] S. R. V and H. Muccini, “Group decision-making in software architecture: A study on industrial practices,” *Information and Software Technology*, vol. 101, pp. 51–63, Sep. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0950584918300740

[52] M. L. Drury-Grogan, K. Conboy, and T. Acton, “Examining decision characteristics & challenges for agile software development,” *Journal of Systems and Software*, vol. 131, pp. 248–265, Sep. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0164121217301103

[53] A. Tang, M. Babar, I. Gorton, and J. Han, “A Survey of the Use and Documentation of Architecture Design Rationale,” in *5th Working IEEE/IFIP Conference on Software Architecture (WICSA’05)*, Nov. 2005, pp. 89–98.

[54] Ralph L Keeney, “Creativity in Decision Making with Value-Focused Thinking,” 1994.

[55] G. Parnell and P. West, “KR23 Value-Focused Systems Decision Making,” *INCOSE International Symposium*, vol. 18, pp. 1685–1699, Jun. 2008.

[56] Gregory S. Parnell, “Chapter 19 Value-Focused Thinking.”

[57] C. W. Kirkwood, *Strategic decision making: multiobjective decision analysis with spreadsheets*. Belmont: Duxbury Press, 1997.

[58] David D. Walden, Garry J. Roedler, Kevin J. Forsberg, R. Douglas Hamelin, and Thomas M. Shortell, *INCOSE Systems Engineering Handbook*, 4th ed. Wiley, 2015.

[59] M. LaSorda, “Applying Model-based Systems Engineering to Architecture Optimization and Selection During System Acquisition,” Ph.D., Colorado State University, United States – Colorado, 2018, iSBN: 9780438789760. [Online]. Available: http://www.proquest.com/docview/2170601880/abstract/8CB3A7CF008B49EEPQ/1

[60] M. Patil, “Model Based System Engineering (MBSE) For Accelerating Software Development Cycle,” p. 18, 2015.

[61] T. G. Gebreegziabher, L. Qiao, Y. Qie, and N. Cai, “A Model-Based Method for Assisting Decision Making Process in Product Development,” in *2017 5th International Conference on Enterprise Systems (ES)*, Sep. 2017, pp. 93–98, iSSN: 2572-6609.

[62] "SysML FAQ: What is the relation between SysML and UML?" [Online]. Available: https://sysmlforum.com/sysml-faq//labels//sysml-faq/what-is-relation-between-sysml-and-uml.html

[63] "SysML FAQ: What is SysML?, Who created SysML?, ..." [Online]. Available: https://sysml.org/sysml-faq/index.html

[64] "Types of Models SEBoK." [Online]. Available: https://www.sebokwiki.org/wiki/Types_of_Models

[65] "MagicDraw - CATIA - Dassault Systèmes®." [Online]. Available: https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/

[66] "Hourly wage for Software Engineer I | Salary.com." [Online]. Available: https://www.salary.com/research/salary/benchmark/software-engineer-i-hourly-wages

[67] "Logical Decisions, Software for more effective decisions." [Online]. Available: http://www.logicaldecisionsshop.com/catalog/

[68] "Definition of Small And Midsize Business (SMB) - Gartner Information Technology Glossary." [Online]. Available: https://www.gartner.com/en/information-technology/glossary/smbs-small-and-midsize-businesses

[69] R. L. KEENEY and R. L. Keeney, *Value-Focused Thinking: A Path to Creative Decisionmaking*. Harvard University Press, Jun. 2009.

[70] "EC2 On-Demand Instance Pricing – Amazon Web Services." [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/

[71] "High-Performance Block Storage– Amazon EBS Pricing – Amazon Web Services." [Online]. Available: https://aws.amazon.com/ebs/pricing/

[72] "Network Traffic Distribution—Elastic Load Balancing Pricing –Amazon Web Services." [Online]. Available: https://aws.amazon.com/elasticloadbalancing/pricing/

[73] "Logically Isolated Virtual Network - Amazon VPC Pricing - Amazon Web Services." [Online]. Available: https://aws.amazon.com/vpc/pricing/

[74] "Amazon RDS for PostgreSQL Pricing – Amazon Web Services." [Online]. Available: https://aws.amazon.com/rds/postgresql/pricing/

[75] "Amazon OpenSearch Service (successor to Amazon Elasticsearch Service) Pricing – Amazon Web Services." [Online]. Available: https://aws.amazon.com/opensearch-service/pricing/

[76] "Fully Managed Graph Database - Amazon Neptune Pricing - Amazon Web Services." [Online]. Available: https://aws.amazon.com/neptune/pricing/

[77] "Amazon S3 Simple Storage Service Pricing - Amazon Web Services." [Online]. Available: https://aws.amazon.com/s3/pricing/

Curriculum Vitae

Kelly Fitzpatrick graduated from Sabino High School, Arizona, in 2009. She received her Associates of Arts Liberal Arts from Pima Community College in 2012. She received her Bachelor of Science with a duel major in Computer Science and Mathematics with an Emphasis in Computer Science from the University of Arizona in 2015. She received a scholarship to attend George Mason University starting Fall 2020 for a Master of Science in Systems Engineering.