USA GM: An Analysis and Visualization of I.T. Expenditures Across A Government Agency

By

Wesley Chin

Nolan Costigan

Jiwan Hwang

Abinash Mohanty

Alfonso Panlilio

PROJECT REPORT

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Data Analytics Engineering at the Volgenau School of Engineering of George Mason University

Spring 2019

Table of Contents

Table of Contents 2

Abstract 4

Introduction 5

- Background and Rationale 5
- USAGM 6
- Project Objectives 7
- Problem Definition 7
- Solution Space 7
- Product Vision 8

Design 9

- Architecture 9
- Web Application 10
- Classifier Integration 13
- Dashboard 13

Data 14

- Background 14
- Mock Data Process 14
- Other Data Sources 15
- Data Object Model 15
- Data Flow 17

Analytics and Algorithms 17

- Classification Analytics 17

Visualization 24

- Background 24
- Pre-visualization process using Power BI 24
- TBM Cost Analysis Dashboard 25
  - Use Case: Outlier Spending 27

Time Series and Entity Analysis 27
Use Case: Tracking Expenses Through Time 30
Use Case: Tracking Expenses by Entity 30
Unit / License Usage Dashboard 31
Use Case: Unused Licenses 31
Shadow IT Dashboard 32
Use Case: Products with Multiple Uses 33
Summary 33
Future Work 33
Azure Active Directory 33
Python-to-C# Communication Redesign 34
TBM Taxonomy 34
Appendix A - Technical Content 35
Classifier Web Application 35
Classifier Python Application 37
Approve Web Application Page (in ITInvestmentsController.cs) 42
ITInvestment Object (ITInvestment.cs) 43
ITFunction Object (ITFunction.cs) 45
Appendix B - Agile Development 47
Appendix C - Common References 49

Abstract

In 2014, the Federal Information Technology Acquisition Reform Act (FITARA) was enacted to install a framework for governing the management of Information Technology (IT) solutions employed within the federal government. Under the governance of the United States House Committee on Oversight and Government Reform (OGR), each federal agency is required to provide consolidated IT expenditure reports that cover overarching mission goals. The goal of this legislation was to increase spending transparency and identify potential redundancies in spending. The United States Agency for Global Media (USAGM) currently organizes its IT expenditures through the use of a shared spreadsheet as its FITARA solution.

This paper details the design and creation of a web application solution that addresses the requirements of FITARA, and assists USAGM in identifying potentially redundant spending. The web application consolidates and centralizes data pieces required by OGR, identifies redundancies in IT spending, automates expenditure visualization, and allows real-time visualization filtering and organization.

1 Introduction

1.1 Background and Rationale

Federal IT Procurement is a complex process that requires collaboration from multiple stakeholders. These include end users, program managers, contracting officers, and senior executives. One challenge of IT procurement is agencies must weigh the benefits of allowing end users to make IT decisions vs centrally procuring hardware and software to reduce cost. End users typically know the technology required to achieve mission objectives but centrally procuring technology saves money and time. Agency CIO's are well positioned to understand technology requirements as well as manage the sometimes convoluted world of federal IT budgeting and procurement.

The Federal IT Acquisition Reform Act (FITARA)[1] is a US legislation passed in 2014 which, among other things, put the CIOs of federal agencies in charge of IT investments. The legislation was enacted to respond to several federal IT challenges. These include duplicate IT spending, understanding of cost and & performance of IT spending, lack of visibility into IT spending, and inability to benchmark IT performance across federal and private sector counterparts. With the increase in responsibility and control given to federal CIOs by FITARA comes accountability counterpart. Federal agencies have to provide the Office of Management and Budget with an IT expenditure report which shows how IT properly aligns with the mission goals of the agency.

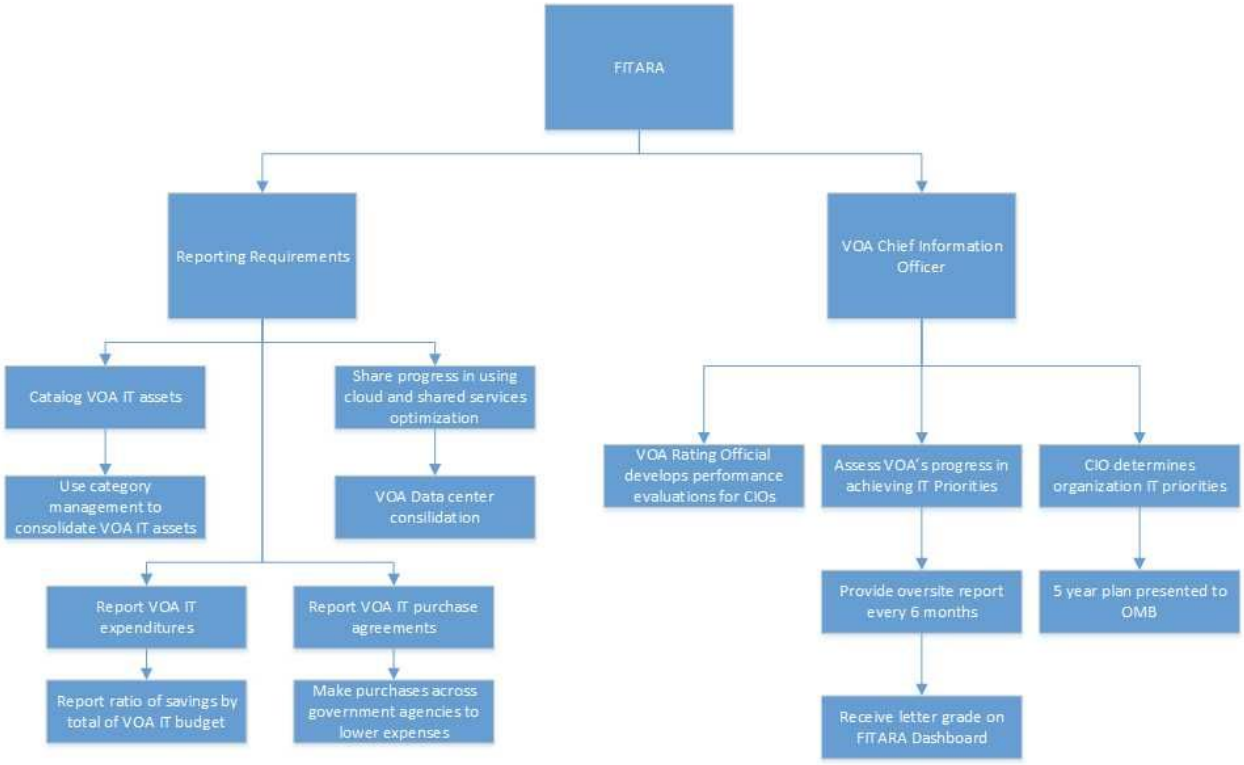


Figure 1.1.1. FITARA organizational requirements

[1] FITARA Summary. Apptio. 2019. https://www.apptio.com/fitara-summary.

In order for federal agencies to adhere to this legislation a proper way of categorizing IT investments and expenditures has to be set. The Technology Business Management Council (TBM)[2] is an organization composed of CIO, CTO, and CFO members across different industries. The goal of TBM is to create and promote the best practice for running IT. TBM provides a value management framework for IT expenditures with the goal of properly aligning these costs with the business and finance of the organization. This taxonomy is the generally accepted way of reporting IT costs and other metrics. The `TBM framework can be leveraged to address the needs of FITARA as well as provide the CIO with proper insight into IT investments for better decision making.

The figure below shows the taxonomy[3] used to categorize IT costs from the IT, business, and Finance perspectives.

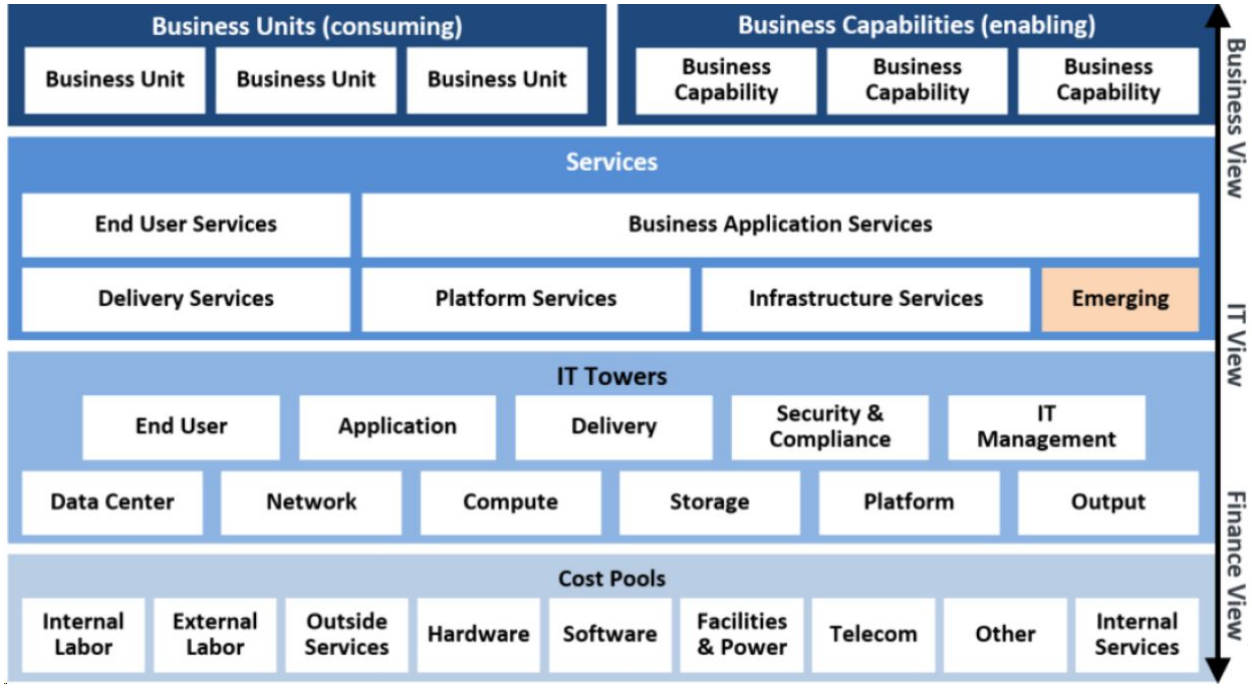


Figure 1.1.2. The TBM taxonomy provides a standard set of categories for costs and other metrics

1.2 USAGM

USAGM, like many agencies, tends to be fairly decentralized, in that they have multiple centers creating requirements and procuring IT hardware and software separately. This creates two prominent challenges. The first is it increases the chances of duplicate spending as well as purchases of hardware and software that go unused. This is typically referred to as "shadow IT costs." The second is the inability to report on these expenses as per FITARA compliance.

Currently, USAGM organizes the entirety of its IT expenditures through the sharing of a single spreadsheet document, aptly nicknamed the MOAS, or Mother of All Spreadsheets. For FITARA reporting requirements, this MOAS is periodically scraped and the data is visualized. TBM offers a best practice framework for categorizing IT costs which align with the finance and business.

[2] TBM Council. https://www.tbmcouncil.org/
[3] TBM Taxonomy. TBM Council. 2 Nov 2018. pp 1.
https://www.tbmcouncil.org/sites/default/files/tbmc_tbm_taxonomy_3.0.pdf

This project is an opportunity to make the MOAS process less burdensome and clunky, and identify Shadow IT, the unknown redundant IT expenditures, within the organization.

1.3 Project Objectives

There are two primary objectives for our project:

First, we will establish a prototype for moving away from the current single-spreadsheet MOAS information model. The prototype should simplify user input, maintain a centralized data storage, and automate certain dashboard visualizations.

Secondly, we will automate the identification of shadow IT through classification algorithms. With this, the prototype will be able to suggest alternative solutions when a user inputs a purchase order and prevent multiple purchases for items that have similar or duplicate functionality.

1.4 Problem Definition

Based on the user context and value proposition, we developed the following primary user story to guide our project:

"As a User, I want to ingest and aggregate investment expenditures so that I can monitor IT expenditures comprehensively, create dashboard visualizations of necessary information, and identify solution areas of redundancy and noncompliance while maintaining the status quo of processes for lower level expense reporting"

1.5 Solution Space

There are four core areas where our application solution brings value to USA GM.

Our IT function classifier helps client users implement solutions from other subdivisions within their agency. By suggesting approved technologies at time of input, the classifier can help reduce redundant purchases before they happen. For example: If a group needs additional licenses of a statistical packaging software, they can enter this information into the classifier and it will recognize that another group in the agency has already purchased licenses for a product that meets their needs. What's more- our dashboard (to be discussed later) will indicate that the existing licenses are not being fully utilized and may have unused seats. This may negate the need for a new purchase in the first place- thus saving the agency time and money.

Our centralized storage solution transitions USA GM away from a MOAS informational model. Centralized storage assures that the expenditure information is not subject to file transfer or hard drive failure risks. Centralized storage and indexing can group expenditures by function and help identify where redundant spending is occuring. Redundancies and shadow IT can be identified faster and earlier. In addition, it lowers the time it takes for users to update the information. Information can the edited in the cloud and users will not have to download, save a file locally, edit, and upload a file back into the repository. Streamlining any process, such as this one, will save a great amount of time over the course of years.

Our web application provides a centralized method for submitting and tracking IT expenditures. Through the MOAS model, tracking IT expenditures across teams is done through file sharing, which is subject to data inconsistency risks. The web application user interface enforces input types and disallows free-text values. It also streamlines the purchase order submission process. The web application additionally

avoids status quo disruption by providing methods to upload spreadsheets to the database. However, the file must be formatted correctly.

Lastly, our application's visualization dashboard is directly connected to the Azure SQL database. Live updates to the client's expenditures will be reflected in the Power BI dashboard. Responding to live updates means less time spent re-creating and re-formatting outdated charts. Additionally, the dashboard provides filtering and rules for the user so that new charts aren't necessary for capturing new filtering.

1.6 Product Vision

The goal of this product is to help USAGM better manage their IT expenditures and comply with FITARA requirements. It will be structured according to TBM taxonomy to help decision makers follow the councils best practices and recommendations. This product delivers real-time information to organization-wide personnel; from end users, all the way up to C-Level executives.

When designing this product, our team developed a multi-pronged approach by examining the needs of all users in USAGM. The classifier will help end-users and managers better track IT purchases. It allows Finance and Program Managers to have more visibility into the budgeting and planning of the agency. By identifying past spending trends, they can more accurately plan future budgets and report to the Office of Management and Budget (OMB) how they are reducing IT expenditures to comply with FITARA requirements. Lastly, the dashboard delivers real-time information to C-Level executives that allow them to make better, more informed decisions for their organization.

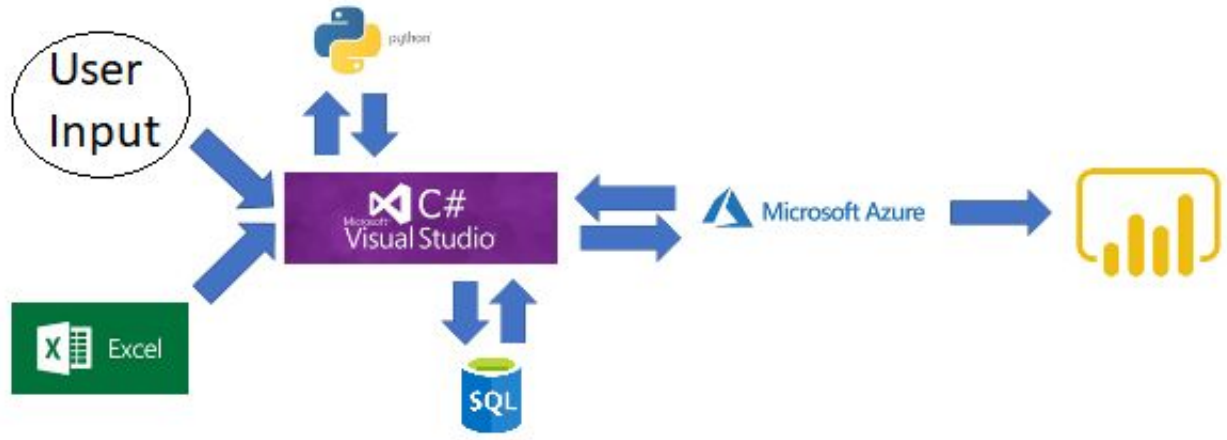


Figure 1.6.1. Our initial design of what our application would entail

The web application would be developed heavily utilizing C#/ASP.NET and hosted on Azure. Users would input data either through the web application or through importing Excel documents. Data would be stored on an Azure SQL Database and Power BI would be drawn from that data source.[4] The python portion of the application represents our classifier operation.

[4] USA GM, a Microsoft customer, specifically asked us to focus on utilizing Microsoft products.

2 Design

2.1 Architecture

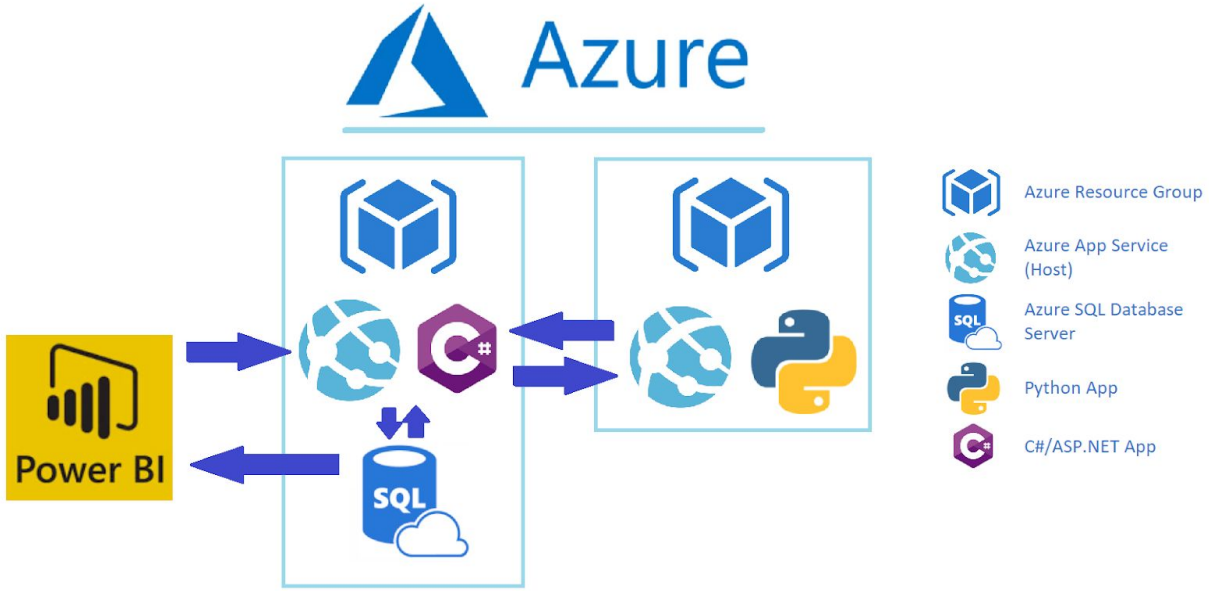


Figure 2.1.1. Technical Overview of the system

The web application was developed utilizing C#/ASP.NET. The primary UI application was hosted on Microsoft Azure[5] in the same resource group as our Azure SQL DB. The python classification web application was hosted in a separate resource group. The Power BI dashboard gets data from the SQL database and is published to the public where it is then displayed on the web application.

The final design mostly kept to our initial design. We had to improvise due to two key reasons. First, we misunderstood Azure hosting architecture. We had to configure Azure resource groups to host our app service. Second, we underestimated the complexity of integrating Python into a C# web application. This integration process is detailed below in Section 2.3 (Classifier Integration).

[5] Microsoft accounts were provided by Practical Solutions, Inc. (PSI). PSI is a contractor for USAGM. We developed our application using PSI's Microsoft subscription.

2.2 Web Application

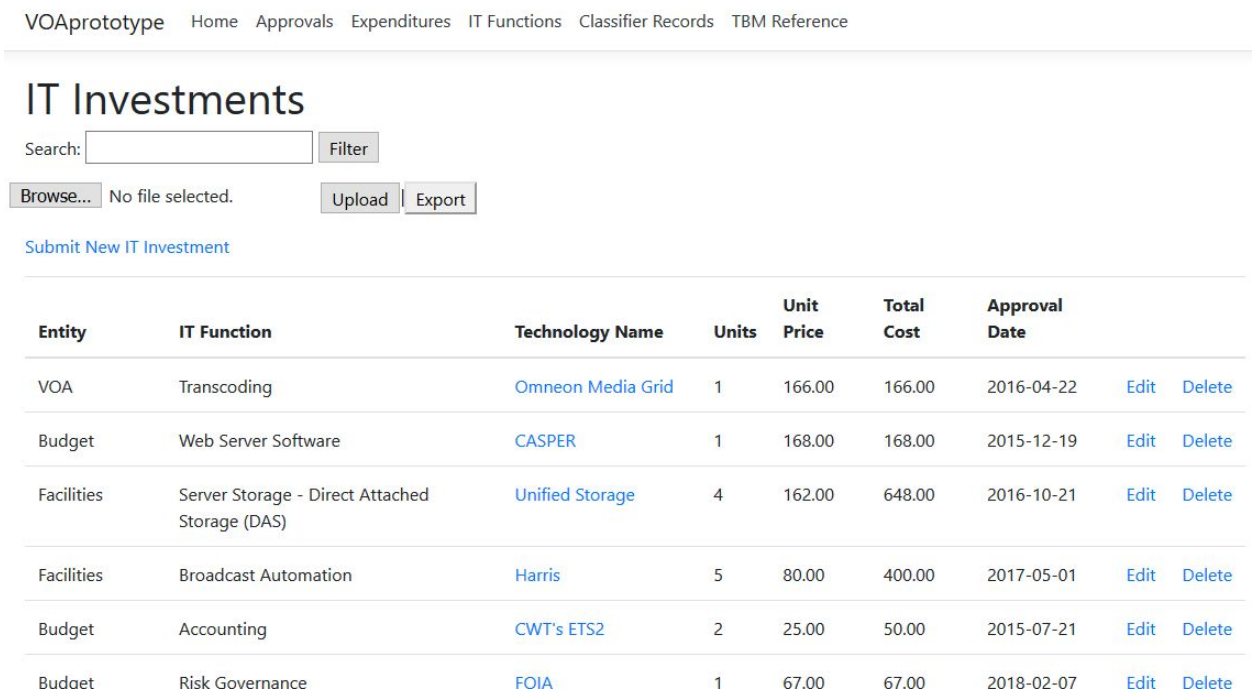


Figure 2.2.1. The web application home page loaded with our mock data (see Section 3)

IT Investments are created, edited, deleted, and approved through the web application's user interface. The web application is coded in C#/ASP.NET Core 2.2 and largely follows the ASP.NET Core MVC tutorial published by Microsoft[6]. The application follows a Model-View-Controller (MVC) architecture[7]. The views (user interfaces) are coded using Microsoft Razor Pages and javascript.

[6] Anderson, Rick, et al. "Create a Web App with ASP.NET Core MVC." *Create a Web App with ASP.NET Core MVC*, Microsoft Corporation, 27 Oct. 2017, docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/?view=aspnetcore-2.2.

[7] Wikipedia contributors. "Model–view–controller." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 20 Apr. 2019. Web. 26 Apr. 2019.

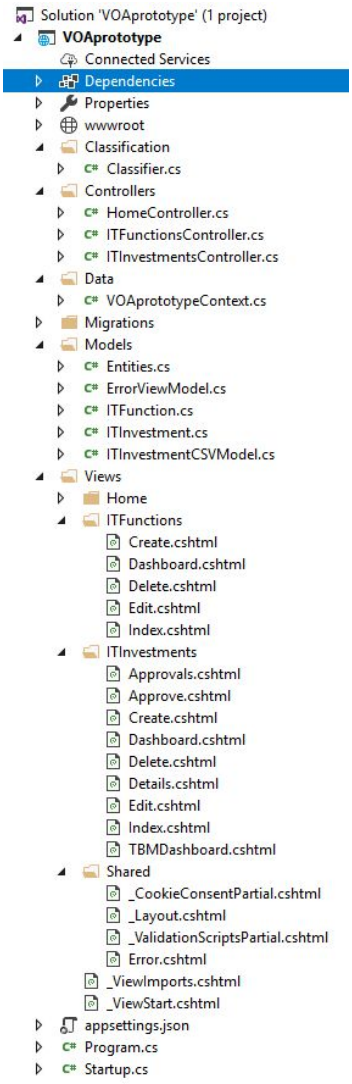


Figure 2.2.2. The web application development tree

Due to privilege constraints (not an Azure admin for Practical Solutions, Inc.), Azure Active Directory (AAD) was not configured for the web application. Without AAD, application security, especially with data of this nature, is lost. However, in order to portray our idea of users and approvers, views were separated to match our vision of privileges.

The Approvals tab and Approve views would only be viewable to approvers.

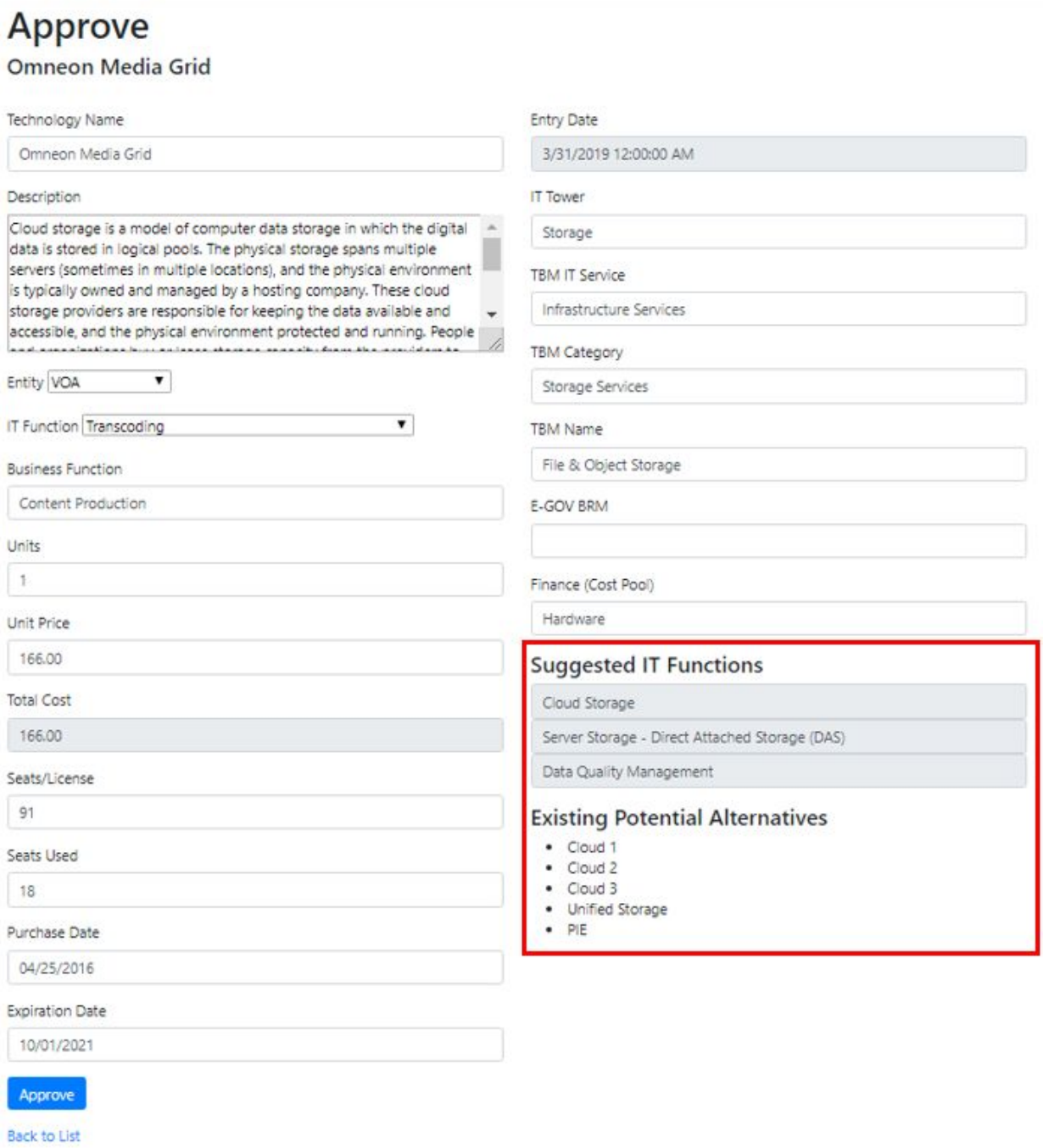


Figure 2.2.1. The investment approval page. Outlined is our classifier results.

On the investment approval page, the approver is provided with three suggested IT Functions and a list of investments that have already been purchased and serve similar functionality. This will assist the approver is determining whether this investment is redundant.

2.3 Classifier Integration

Because the classifier and the web application were initially developed separately, there was some development time devoted to integrating the two pieces together.

Initially, we attempted integrating the classifier using C#/ASP.NET's IronPython[8] NuGet package. From their website, "IronPython is an open-source implementation of the Python programming language which is tightly integrated with the .NET Framework. IronPython can use the .NET Framework and Python libraries, and other .NET languages can use Python code just as easily." Our first hurdle was that IronPython does not yet support Python 3 so we attempted redesigning our code around Python 2.7. We were able to run the majority of the classifier, but lost some of Python package sklearn's functionality so we decided to move on to a different alternative.

Our second attempt was to build a Python executable file (.exe) using the pyinstaller[9] Python package. "PyInstaller "freezes" Python applications into stand-alone executables." After a lot of troubleshooting and debugging, we learned that PyInstaller is not 100% compatible with all Python packages. Specifically, some of nltk's sub-packages were incompatible.

As our final option, we settled on writing a Python web application that could be communicated with via HTTP Requests. Although building a Python web application to host on Azure was our final solution, this design is fairly inefficient, insecure, and overall follows poor design.

Re-designing the classification communication architecture is a priority candidate for improvement.

2.4 Dashboard

The dashboard embedding is also limited by the lack of Azure AD privileges. To circumvent these issues, we embedded our dashboard using Power BI web application's Publish to Web functionality.

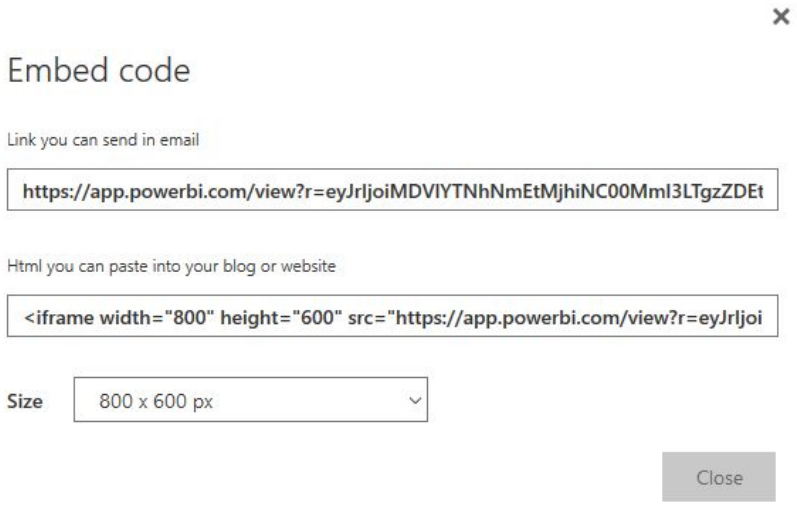


Figure 2.4.1. Power BI application's public embedding UI

[8] IronLanguages, IronPython2 (2019), GitHub repository, https://github.com/IronLanguages/ironpython2
[9] PyInstaller, PyInstaller (2019), GitHub repository, https://github.com/pyinstaller/pyinstaller

With this public embedding, the dashboard's data source also has security concerns since the page had to be made publicly available. The Power BI application's public embedding UI provides an HTML code snippet. These snippets were copied and pasted to our web application views.

3 Data

3.1 Background

Since we are building a full-fledged prototype for automated FITARA dashboard visualizations, the dataset does not yet exist. As part of our project, we designed the data input schema for our system. In order to fulfill all of FITARA dashboard visualization requirements, the dataset must contain a minimal amount of dimensions. A lot of the proposed dimensions coincide with the TBM taxonomy. Said in an earlier section TBM provides a framework for reporting IT expenditures. The data schema is largely based on the TBM taxonomy as well as other metrics we may deem useful for providing insight.

3.2 Mock Data Process

Initial process of gathering information from the mock data was through back and forth discussions with the client. VOA provided guidance as to what type of structure the data should be in. This guidance came with a recommendation to perform research on FITARA and TBM taxonomy. The legislation and the taxonomy served as a baseline and foundation for creating the mock data.

2019 Service Portfolios (based on TBM ver.3) mapped to FISMA Boundaries

OMB Capital Planning	TBM Product & Service Portfolio (Type)	TBM Service (Categor	TBM Service (Name	Budget	Supporting Services (Sut	VOA-TSI Current	VOA-TSI Current So	VOA-TSI Curre	FISMA Systems	Planned Replacements
Part 1 - Mission	Platform (TV)	Application Services	Content Management	TSI	File Content Transfer	Peach AWS	Aspera		TSI Content Prod. - File Transfer	
			Content Management	TSI	Paid feeds (Video, Audio,	Reuters	AP		TSI Content Prod.- MAM	
			Content Management	TSI	Content Ingest tools	Dalet Brio	NCH Software		TSI Content Prod.- MAM	
			Content Management	TSI	Video Editing, News Management-Production	Dalet Plus			TSI Content Prod.- MAM	
			Content Management	TSI	Media Transcoding	Rhozet	Limigito		TSI Content Prod.- MAM	
		Application Services	Content Management	VOA	TV Studio Operation	Vinton Robotics	Control	Audio	N/a (Not on WAN)	
			Content Management	VOA	TV Production Switchers	Sony MVS Series	Broadcast Pix		N/a (Not on WAN)	
			Content Management	VOA	Content Ingest tools	Streambox	Pesa C22	Teradek Core	VOA SBS - TVProduction	
			Content Management	VOA	TV Production Switchers	Newtek Tricaster			VOA SBS - TVMC & Broadcast Core	
			Content Management	VOA	TV Automation	Harris			VOA SBS - TVMC & Broadcast Cor	GVG iTX, Dell Isilon,
			Content Management	VOA	TV Graphics	Chyron/Camio	ORAD TD Control	Chyron iSQ	VOA SBS - TVMC & Broadcast Cor	GVG Vertigo (CG)
			Content Management	VOA	Video Editing	Avid/ISIS			VOA SBS - TVProduction	
					Audio Editing, News		Dalet RSHD			

Figure 3.2.1. The initial spreadsheet data provided (without costs)

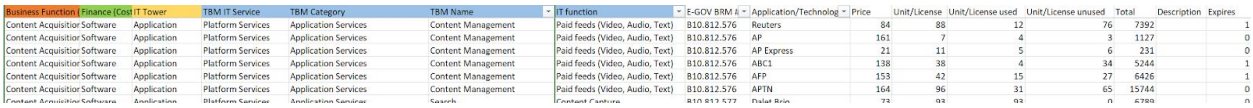
The client provided us with a Microsoft Excel spreadsheet containing existing IT investments which was partially categorized using TBM taxonomy. This served as a starting point for generating the mock data. After more research on TBM, the data was then categorized correctly using the latest taxonomy. Columns containing the unit count, price, and description in this process to add more info that would be pertinent in analyzing investment data.

TBM Name	IT function	E-GOV BRM #	Application/Technolog	Price	Unit/License	Unit/License used	Unit/License unused	Total	Description
Content Management	Paid feeds (Video, Audio, Text)	B10.812.576	Reuters	43	43	29	14	1849	
Content Management	Paid feeds (Video, Audio, Text)	B10.812.576	AP	151	78	27	51	11778	
Content Management	Paid feeds (Video, Audio, Text)	B10.812.576	AP Express	169	78	73	5	13182	
Content Management	Paid feeds (Video, Audio, Text)	B10.812.576	ABC1	80	32	32	0	2560	
Content Management	Paid feeds (Video, Audio, Text)	B10.812.576	AFP	26	46	31	15	1196	

Figure 3.2.2. Our first iteration for designing the mock data

After the data was fitted to match the taxonomy other elements were added such as randomized cost, units bought, and units used. These were added to have the data properly reflect actual IT costs. Other additions such as creating expiration dates were added to provide proper insight when constructing the dashboard.After some additions to the data to diversify and make it properly reflect a complete IT environment, the data was sent to the client for approval. The first six columns align with the TBM

taxonomy and were added as the main structure for categorizing the different types of IT technologies. These align with the three main parts of the TBM taxonomy which are IT, Business, and Finance.



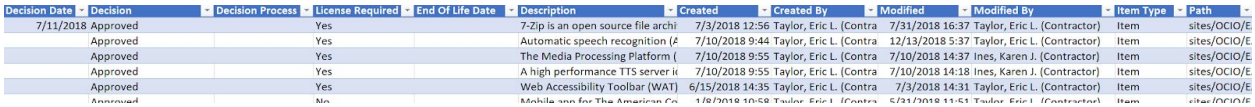
Business Function (	Finance (Cost	IT Tower	TBM IT Service	TBM Category	TBM Name	IT function	E-GOV BRM Ti	Application/Technolog	Price	Unit/License	Unit/License used	Unit/License unused	Total	Description	Expires
Content Acquisitior	Software	Application	Platform Services	Application Services	Content Management	Paid feeds (Video, Audio, Text)	B10.812.576	Reuters	84	88	12	76	7392		1
Content Acquisitior	Software	Application	Platform Services	Application Services	Content Management	Paid feeds (Video, Audio, Text)	B10.812.576	AP	161	7	4	3	1127		0
Content Acquisitior	Software	Application	Platform Services	Application Services	Content Management	Paid feeds (Video, Audio, Text)	B10.812.576	AP Express	21	11	5	6	231		0
Content Acquisitior	Software	Application	Platform Services	Application Services	Content Management	Paid feeds (Video, Audio, Text)	B10.812.576	ABC1	138	38	4	34	5244		1
Content Acquisitior	Software	Application	Platform Services	Application Services	Content Management	Paid feeds (Video, Audio, Text)	B10.812.576	AFP	153	42	15	27	6426		1
Content Acquisitior	Software	Application	Platform Services	Application Services	Content Management	Paid feeds (Video, Audio, Text)	B10.812.576	APTN	164	96	31	65	15744		0
Content Acquisitior	Software	Application	Platform Services	Application Services	Search	Content Capture	B10.812.572	Dalet Brio	73	93	93	0	6789		0

Figure 3.2.3. Our second iteration for designing the mock data

After some additions to the data to diversify and make it properly reflect a complete IT environment, the data was sent to the client for approval.

3.3 Other Data Sources

Through networking with the client, we were able to consult with Chief Information Officer of the US Small Business Administration Maria Roat about her organization's solution for FITARA regulations.



Decision Date	Decision	Decision Process	License Required	End Of Life Date	Description	Created	Created By	Modified	Modified By	Item Type	Path
7/11/2018	Approved		Yes		7-Zip is an open source file archi	7/3/2018 12:56	Taylor, Eric L. (Contra	7/31/2018 16:37	Taylor, Eric L. (Contractor)	Item	sites/OCIO/E
	Approved		Yes		Automatic speech recognition (A	7/10/2018 9:44	Taylor, Eric L. (Contra	12/13/2018 5:37	Taylor, Eric L. (Contractor)	Item	sites/OCIO/E
	Approved		Yes		The Media Processing Platform (	7/10/2018 9:55	Taylor, Eric L. (Contra	7/10/2018 14:37	Ines, Karen J. (Contractor)	Item	sites/OCIO/E
	Approved		Yes		A high performance TTS server ic	7/10/2018 9:55	Taylor, Eric L. (Contra	7/10/2018 14:18	Ines, Karen J. (Contractor)	Item	sites/OCIO/E
	Approved		Yes		Web Accessibility Toolbar (WAT)	6/15/2018 14:35	Taylor, Eric L. (Contra	7/3/2018 14:31	Taylor, Eric L. (Contractor)	Item	sites/OCIO/E
	Approved		No		Mobile app for The American Co	1/8/2018 10:58	Taylor, Eric L. (Contra	5/31/2018 11:51	Taylor, Eric L. (Contractor)	Item	sites/OCIO/E

Figure 3.3.1. The existing data model used by SBA

Figure 7 shows the model (with a few alterations) that Maria Roat used for tracking IT expenditures within her organization.

3.4 Data Object Model

Once our mock was approved, we designed our web application object model around the mock data schema. Some fields are aggregate (such as cost = units * price) so we simplified for what we needed.

Id	Entity	Business Fu	Finance (Co	IT Tower	IT Function	TBM IT Ser	TBM Categ	TBM Name	E-GOV BRM	Technology	Units	Unit Price	Total Cost	Seats Per L	Seats Used	Description	Entry Date	Purchase D	PurchaseD	Expiration
3	VOA	Content Ac	Software	Application	Multimedi	Platform Se	Application	Content Management		AP Express	5	120	600	90	417	associated	########	########	########	########
4	Administra	Content Ac	Software	Application	Multimedi	Platform Se	Application	Content Management		ABC1	2	105	210	16	1	"The Amer	########	########	########	########
5	Budget	Content Ac	Software	Application	Multimedi	Platform Se	Application	Content Management		AFP	2	50	100	26	5	"Agence Fr	########	########	########	########
6	VOA	Content Ac	Software	Application	Multimedi	Platform Se	Application	Content Management		APTN	2	119	238	69	81	"APTN Nat	########	########	########	########
7	HR	Content Ac	Software	Application	Video capt	Platform Se	Application	Search		Dalet Brio	3	195	585	35	47	"In broadc	########	########	########	########
8	Facilities	Content Ac	Software	Application	Content sh	Business Se	Manufactu	Inventory &	B10.812.57	FTP	5	77	385	15	10		########	########	########	########
9	Administra	Content Ac	Software	Application	Content sh	Business Se	Manufactu	Inventory & Warehou		Peach AWS	4	126	504	16	11	"Amazon V	########	########	########	########
10	Budget	Content Ac	Software	Application	Content sh	Business Se	Manufactu	Inventory & Warehou		DropBox	3	42	126	17	12	"Dropbox i	########	########	########	########
11	VOA	Content Ac	Software	Application	Content sh	Business Se	Manufactu	Service Del	B10.812.576		2	0	0	36	26		########	########	########	########

Figure 3.4.1. An import/export-able CSV representation of the finalized object model

The final purchase order model has these fields:

Field	**Type**	**Description**
Id	integer	The identification denominator for this order
Entity	enumerated values	The entity within USA GM that is making the order
Business Function (LOB)	string	Category based on business function, allocated to each department
Finance (Cost Pool)	string	Category for what type of cost pool is associated with this order (software, hardware, etc.)

IT Tower	string	Category showing the basic building block of IT services
TBM IT Service	enumerated values	Shows IT service provided by the IT investment. Shows the functionality that the investment is used for
TBM Category	enumerated values	Specific category within the IT Service
TBM Name	enumerated values	Further granularity of the TBM Category branching out into individual IT uses under a specific IT Category
E-GOV BRM #	string	Used by the government to categorize similar IT solutions; based on the TBM Service, Category, and Name
Technology Name	string	Name of application/technology/service
Units	integer	The number of licenses/hardware/software
Unit Price	decimal	The price of a single unit
Total Cost	decimal	The total cost of this order
Seats Per License	integer	If this is a license subscription, the number of seats per license purchased
Seats Used	integer	The number of seats from this order that will be using this license
Description	string	A description of the product
Entry Date	DateTime	The original date this order was created
Approval Date	DateTime	The date this order was approved
Purchase Date	DateTime	The date this order was purchased
Expiration Date	DateTime	If this is a license, the date this license expires

3.5 Data Flow

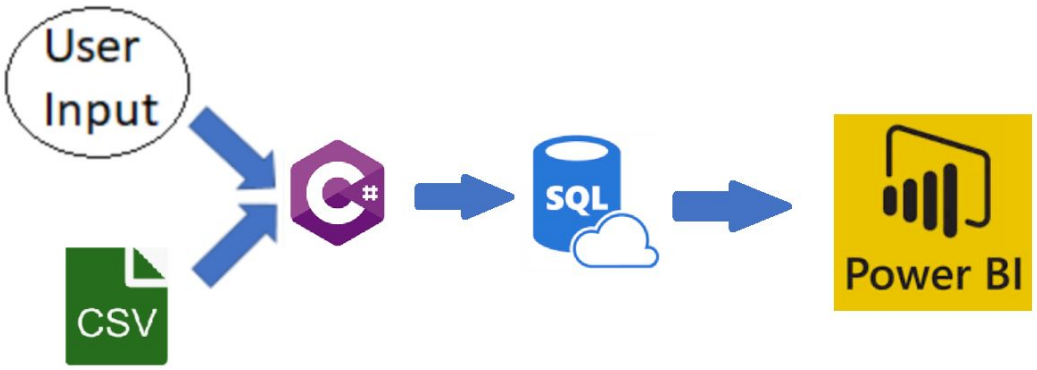


Figure 3.5.1. The flow of data from ingest to visualization

In our designed system, users submit data via the web application using CSV files or through form submission. The web application saves this information to the Azure SQL database. Our Power BI dashboard is directly hooked up to the same Azure SQL database.

4 Analytics and Algorithms

4.1 Classification Analytics

One of the goals of our project is the detection of shadow IT and duplicate/redundant IT technologies. This information can help reduce VOA IT expenditures. For example, when VOA considers the purchase of a particular technology, the organization should be made aware of software that has already been purchased that provides the same functionality as the technology being considered. We were provided data from VOA that contains the names of the various technologies that they have purchased and the support services that these technologies can be classified within. The support services were effectively categories within which similar technologies were classified, for example, the support service: "Video Editing," contained: Dalet Plus, Final CutAvid/ISIS, and Adobe Premier. We decided that our project should include a classifier that will automatically classifies all of VOA's current technology and future technology, allowing the user to see a list of technology that may have similar functionality, in order to achieve the goal of detecting redundancies and duplication.

Since our data is text, we chose to create a text-based classifier. Training a text-based classifier requires a great deal of text and we were not provided any descriptions of the technologies or support services, only names. So, our first challenge was to determine a way to get more information about these categories. After a brainstorming session, we determined the Wikipedia would be a great source for

information on various technologies. There is a open source Wikipedia python package[10], based upon beautiful soup, that makes scraping Wikipedia quite simple. Using only a few commands a user can scrape the content of a Wikipedia page by just referencing the Wikipedia page's title. Also, a user can use the Wikipedia's search engine to get relevant Wikipedia pages that they can scrape. We fed in all of the various names of technologies and support services into the Wikipedia package, scraping both Wikipedia webpages that have titles that match the exact phrase and what appeared as the first search result for each phrase. This strategy of gathering Wikipedia pages was mildly successful, when the title of Wikipedia page matched the exact phrase, the scraped text was most of the time relevant, but when there was no page with the phrase as the title, the first search engine picked out was a relevant page only 50% of the time. In order to guarantee that only relevant Wikipedia pages were being scraped, we hand-audited the pages and chose better Wikipedia pages when an irrelevant page was scraped. In some cases the name of the technology alone was not enough to determine the correct Wikipedia page, these technologies were dropped for our preliminary test.

The text from scraped pages were then loaded into data frames[11]. Typically, any sort of text mining requires the removal of stop words, which are common words that hold little relevance to the topic of a passage of text. Using the nltk package[12], stop words were removed from the scraped text in order to leave more important words. Afterwards, the text of each Wikipedia page was turned into a list of words (unigrams) and a list of 2 words (bigrams). We then chose to transform the list of words using nltk's tfidfvectorizer which provides a vector of numbers representing the importance of each word in the scraped text. Tf-idf (term-frequency inverse document frequency) is a text mining formula which weighs words as more important when they appear frequently in a specific document and weighs words as less important when words appear across documents. The documents in this case are the various Wikipedia pages that were scraped. The vectorized text is required to run the classification algorithms that we were interested in trying: Naïve Bayes, Neural Networks, and Support Vector Machine[13]. We settled in trying these methods after reading that these were best suited for our task.

The training data used to fit the model for all the classification methods were the support services and their corresponding vectorized Wikipedia page text. The test data were the vectorized text of the various technologies' Wikipedia pages. For each of the methods tried, the test data's pairings were compared to the correct support services category. Each method was also tried using bigrams and unigrams.

---

[10] Goldsmith, Jonathan. Wikipedia, commit 2065c568502b19b8634241b47fd96930d1bf948d, Massachusetts Institute of Technology, 11 Nov 2016. *GitHub*, https://github.com/goldsmith/Wikipedia

[11] Wes McKinney. **Data Structures for Statistical Computing in Python**, Proceedings of the 9th Python in Science Conference, 51-56, 2010, http://conference.scipy.org/proceedings/scipy2010/mckinney.html

[12] Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.

[13] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

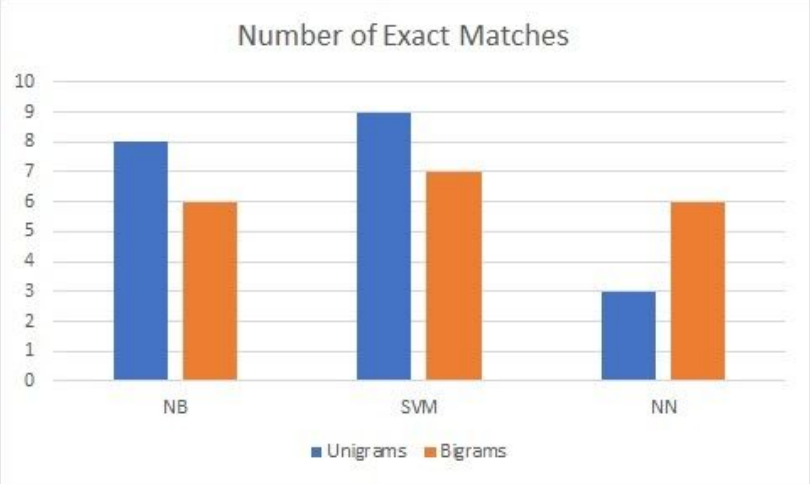


Figure 3.1.1. Number of correct classifications made using each classification method

[The above chart shows how many correct support services classifications each classification method made using bigrams and unigrams. A subset of 30 technologies were classified.]

As shown above, the SVM method using unigrams appears to classify technologies the best, but even this method is only correct 30% percent of the time. This finding lead us to see if the correct support service would be chosen if each method provided its top 3 results.

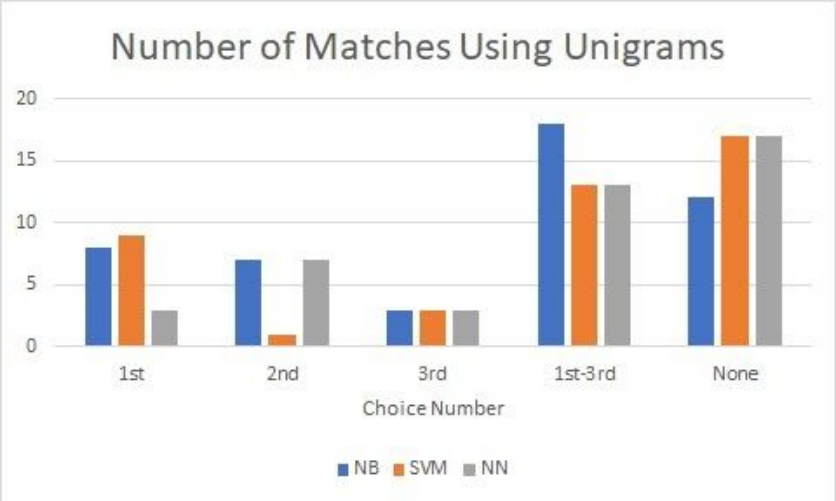


Figure 3.1.2. Number of correct classifications made within the top 3 results of each classification method using unigrams

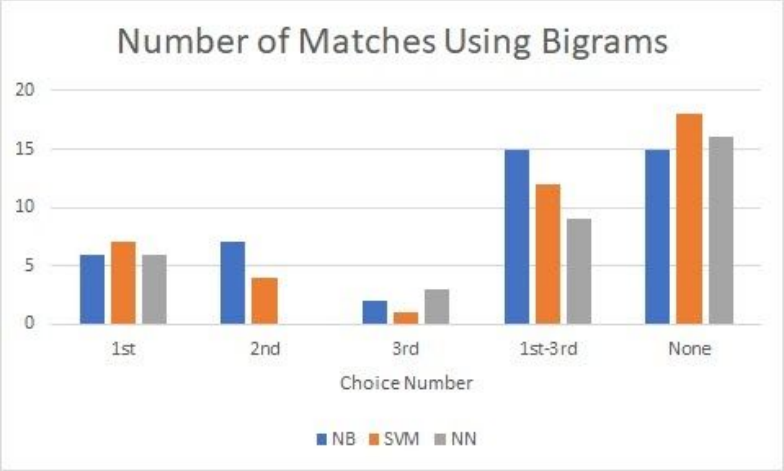


Figure 3.1.3. Number of correct classifications made within the top 3 results of each classification method using bigrams

[1st, 2nd, and 3rd groupings in the previous two charts show the number of times the classification methodology picked the correct support service in the 1st, 2nd, or 3rd choice. The 1st-3rd group shows the number of times the correct support service was selected in the top 3 choices. The none group shows the number of times the classification methodology did not pick the correct support service in the top 3 choices. A subset of 30 technologies were classified.]

Looking at the top 3 results for each method (the 1st-3rd column in the 2 above charts), the naïve bayes algorithm using unigrams appears to be the best. Even so, this classification method was not very successful, with only 50% of the technologies being paired with the correct support service within the top 3 choices. We believed that this was happening because the text in the test data contains a great of deal extraneous unimportant text that is on most Wikipedia pages. For example, many Wikipedia pages have a great deal of unimportant information on the history of the technology, which is probably not relevant for our text mining purposes. The extraneous text could be hampering the effectiveness of tfidf, much like stop words. We think it is important to figure out a way to get more specific descriptions of VOA's technologies.

We believed that user inputted descriptions of VOA's technologies would be classified substantially better than scrapped wikipedia pages. We tested this theory using Small Business Administration data. We received SBA's data after meeting with the Small Business Administration's chief technology officer. Unlike VOA, SBM has detailed data on technology purchases; each technology purchased by SBM has a user inputted description and is classified into a service standard category (effectively SBM's version of support service category). Using the technique described earlier, we scraped and vectorized wikipedia text corresponding to each SBA service standard category. We also vectorized the user descriptions of each technology. We then trained the model using the scraped wikipedia text and their corresponding SBA services standard categories. We tested the model using the inputted descriptions of each purchased technology. Initially, it appeared as though user inputted description did not perform as well as anticipated.

Figure 3.1.4. Number of correct classifications made within the top 3 results of the naive bayes method using unigrams

[1st, 2nd, and 3rd groupings of the previous chart show the number of times the naive bayes classification methodology picked the correct service standard in the 1st, 2nd, or 3rd choice. The 1st-3rd group shows the number of times the correct service standard was selected in the top 3 choices. The none group shows the number of times the classification methodology did not pick the correct support service in the top 3 choices. 489 SBA technologies were classified.]

However, it became clear from further examination of the results that many of the user descriptions were being correctly classified but not into the exact category picked by SBA. The SBA service standard categories list a was far more comprehensive list of IT functions than the service standard list from VOA. So the classifier was often picking relevant categories out of the list, but not the exact one picked by SBA. A hand audit of the results showed that 58% of the time the classifier picked at least one relevant category out of the three categories chosen.

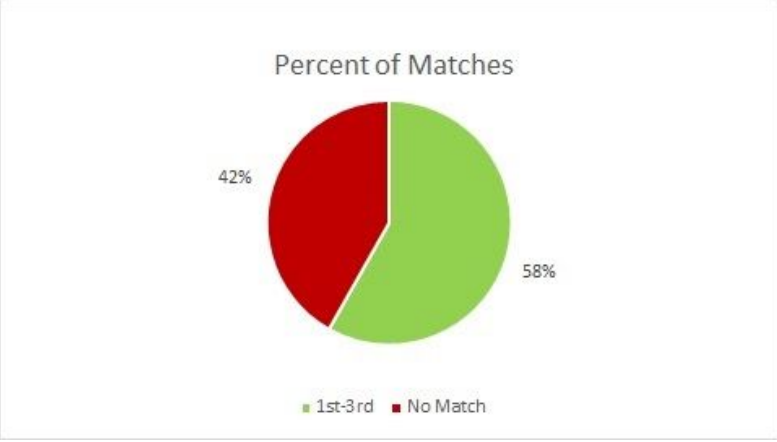
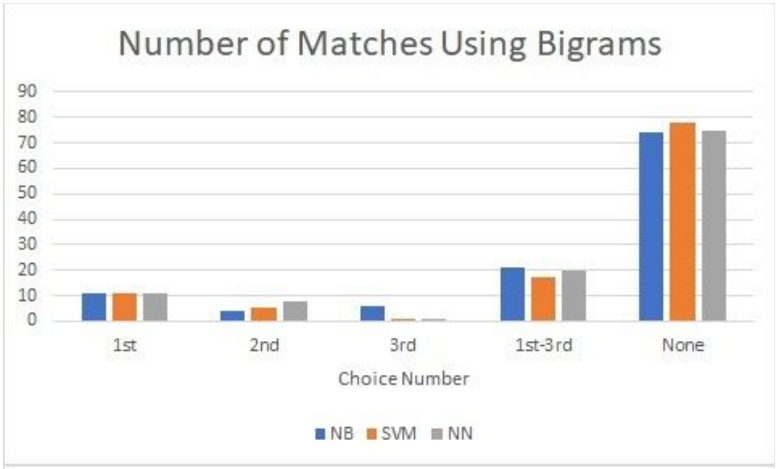


Figure 3.1.5. Percentage of relevant SBA technology classifications made by the naive bayes algorithm using unigrams

[The above chart show the percentage of the times, determined by a hand-audit, that the naive bayes classification methodology picked a relevant SBA service standard category]

When one considers that that the length of the typical user descriptions provided was one sentence, 58% accuracy was quite good. The more comprehensive list of IT function categories provided by SBA also made it easier to identify shadow IT, as technologies were being assigned IT functionalities beyond those identified by SBA. Our success in using SBA's user descriptions made it clear that we could have VOA provide its own descriptions of any additional technologies purchased in the future that it would like to classify, instead of scraping additional Wikipedia pages.

After analyzing these results, we decided that we should classify VOA's technologies into the SBA's service standard categories, since their categories covered IT technology much more comprehensively. To do this we created a column in our mock data that contains each technologies' matching SBA service standard category, using the original VOA support service category as guidance; this new column was titled IT function. We then trained our model using the IT functions and their corresponding scraped Wikipedia pages and we tested our model on VOA's technologies' scraped wikipedia text.



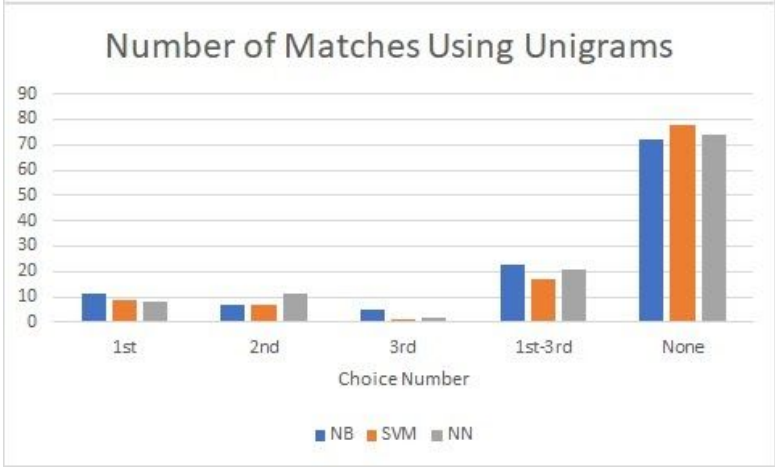


Figure 3.1.6. Number of correct classifications made within the top 3 results of each classification method using unigrams

Figure 3.1.7. Number of correct classifications made within the top 3 results of each classification method using bigrams

[1st, 2nd, and 3rd groupings in the previous 2 charts show the number of times the classification methodology picked the correct IT Function in the 1st, 2nd, or 3rd choice. The 1st-3rd group shows the number of times the correct IT function was selected in the top 3 choices. There were 96 VOA technologies classified in total.]

Looking at exact matches within the top 3 choices of each classification method, the naive bayes using unigrams performed the best. But once again a hand audit was needed to see how many times at least one relevant IT function was chosen out of the top 3 choices.

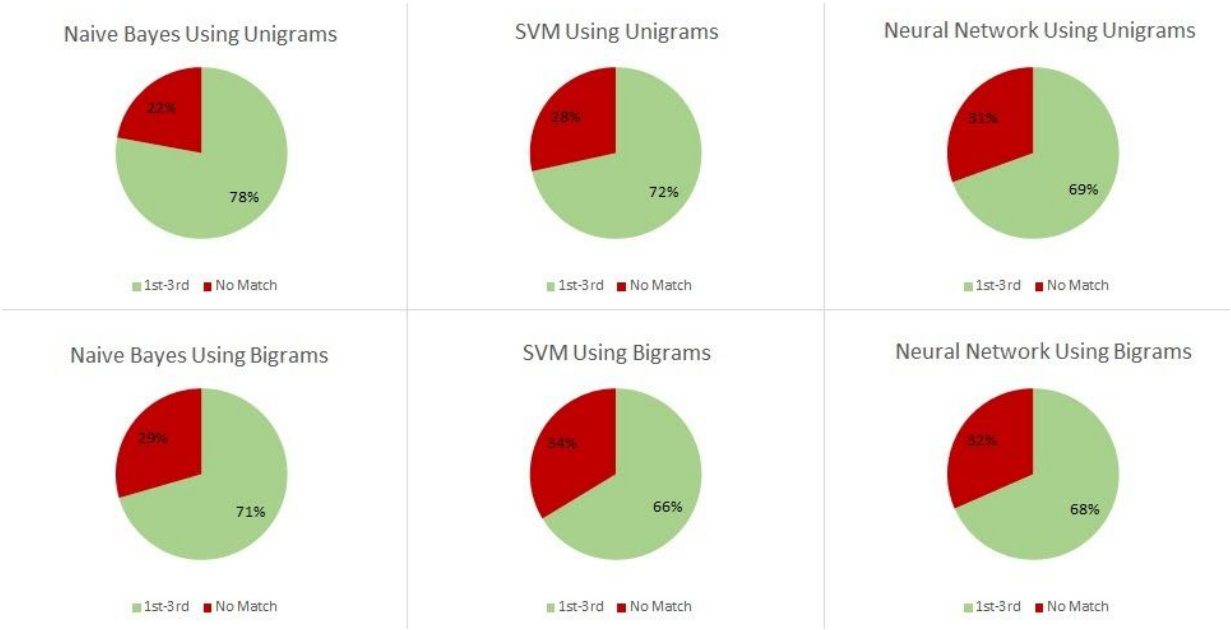


Figure 3.1.8. Percent of relevant VOA technology classifications made by the naive bayes method using unigrams

[Previous graphic shows the percentage of times, determined through a hand audit, that each classification method picked a relevant IT function in the 1st, 2nd, or 3rd choice. There were 96 VOA technologies classified.]

Since naive bayes using unigrams had the highest percentage of relevant IT functions chosen, it was determined to be the best of the methods tried. After deciding on using naive bayes, we implemented the python-based classifier into our web application programmed in C#.

The goal of the classifier within our web application is the detection of IT redundancies and shadow IT. When technology investments are initially inputted into our web application, a user provided description is requested. This user provided description is fed into the the classifier and paired with IT functions. These IT function classifications are shown to the administrators when they approve the addition of technology investments into the database. The administrators are also provided a list of potentially similar technologies within the paired IT functions; with the intention to alert the administrator of any duplicate IT investments. When the administrator approves the addition of a technology investment into the web application's database, the classifier learns, by adding the user inputted description of the technology into the text describing relevant IT functions. This will result in the classifier performing better over time. Additionally, the list of IT functions can be modified by users, allowing VOA to make a

more relevant list of IT functions. Users can add additional IT functions and they can edit the descriptions associated with existing IT functions.

Classifier results are also used in the Power BI reports we have created for VOA. The Shadow IT detection report shows IT expenditures by algorithmically classified IT functions. This allows the user to see costs by relevant IT functions not selected by users, potentially highlighting IT redundancies previously not seen.

5 Visualization

5.1 Background

The main deliverable of this project is a set of interactive dashboards created in Microsoft Power BI. The dashboards deliver real-time data and reports to allow the CIO and other USAGM staff track their IT expenditures. The mock data set for our system is designed based on TBM taxonomy and includes expenditure information about the agency as a whole. Four of our interactive dashboards were built to analyze cost and usage in terms of TBM taxonomy. One additional dashboard was built to analyze cost by computer classified IT function. The five dashboards were: TBM Cost Analysis, Time Series Analysis, Entity Analysis, Unit/License, and shadow IT detection.

Dashboard versions 1.0 and 2.0 were used as brainstorming tools to better understand where we could create value for the agency. Fully-functioning dashboards 3.0 were made after the mock data was approved. Version 3.0 comprised of dashboards similar to CostPerform- a software program the Small Business Administration uses for TBM analysis. However, upon demo of version 3.0, our client suggested dashboards similar to reports that can be shown to executives. This meant simplifying the user interface- reducing the amount of windows and enlarging key graphs within the dashboard. The result was dashboard version 4.0. In accordance with the client's suggestions, standardized drop boxes were adopted throughout the dashboards and most summary tables were removed except for Licence/Unit Usage dashboard. Certain graphs were moved to more prominent space on the report and some were enlarged to show key information more quickly.

A Power BI report and dashboard have different concepts in the Power BI ecosystem. Whole pages of Power BI reports were used as Power BI dashboards by using the Pin Live Page function on the Power BI portal. This enabled the dashboards to be user interactive. If the user wants to bring a specific visual into a dashboard, she can do it by using the Pin Visual function, which is located at top right side of each visual on the Power BI portal.

5.2 Pre-visualization process using Power BI

Microsoft Power BI is a great tool for interactive visualization. It is easy to learn and utilize Power BI's interface because it is very similar to the interfaces of Excel and Access. For example, a few columns such as Total Costs and Availability, were created using The Data Analysis Expressions (DAX) language, which is very similar to Excel formulas. The data schema on Power BI desktop is based off of raw data (units, unit Price, expiration date) that was created in the database.

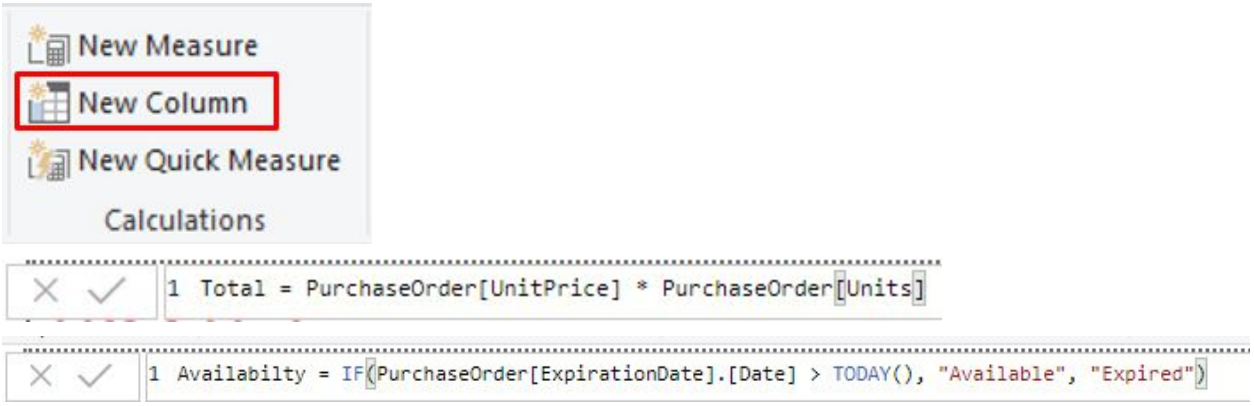


Figure 5.2.1. Creating a New Column using DAX language on Power BI

To build up effective dashboards, Power BI visuals, such as Slicer (List), Card, Multi-row card, Table, Clustered column chart, Line chart, and Pie Chart, were used for each dashboard's own purpose.

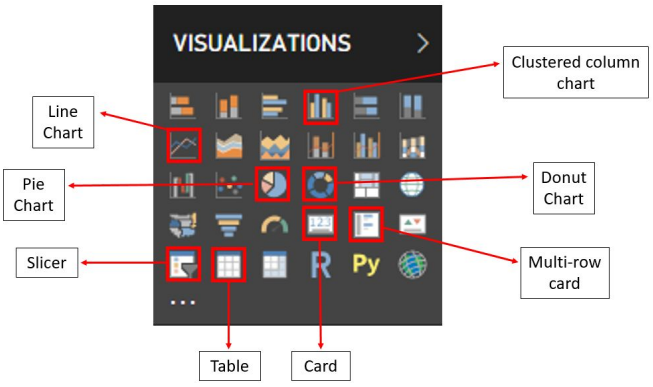


Figure 5.2.2. Power BI visuals used for dashboard 3.0

5.3 TBM Cost Analysis Dashboard

The TBM Cost Analysis dashboard 3.0 is designed based upon TBM taxonomy hierarchy. First, we created the three main layers of TBM taxonomy: cost pool, IT tower and business layer(TBM IT service), and their sub-categories using the slicer function (list), bar chart, and pie chart on Power BI. Each IT application and product is assigned to each of these layers and their categories. When users select one of the layers or categories on the slicers, all the graphs, total cost section, and summary table on the dashboard automatically change and show users relevant information. In addition, users can click plots directly to figure out the portion of a specific layer or category.

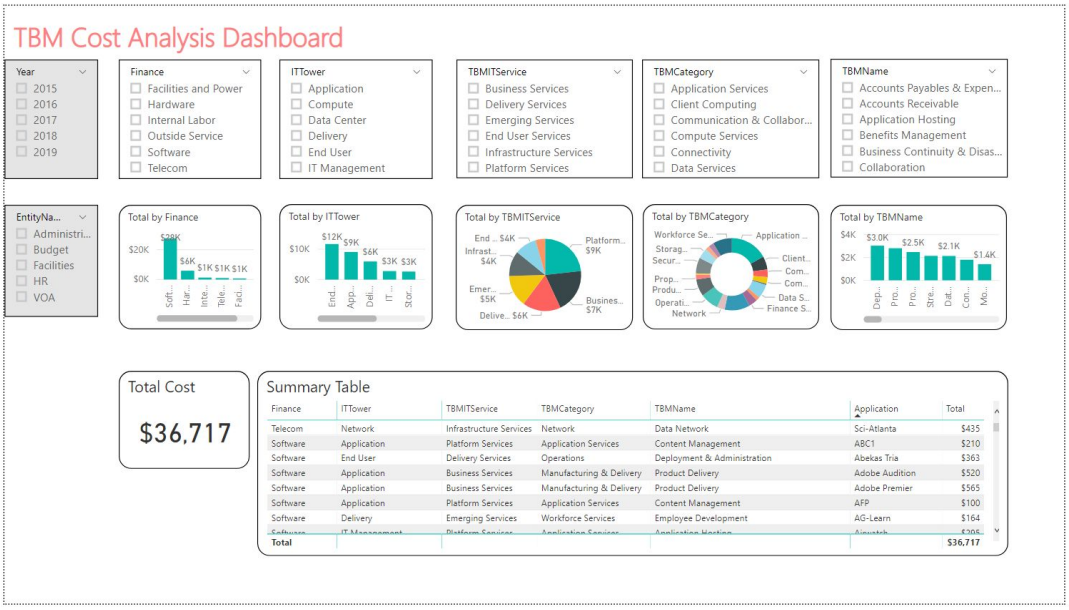


Figure 5.3.1. Cost Analysis Dashboard 3.0 consists of slicers, graph, card, and table functions about TBM Taxonomy

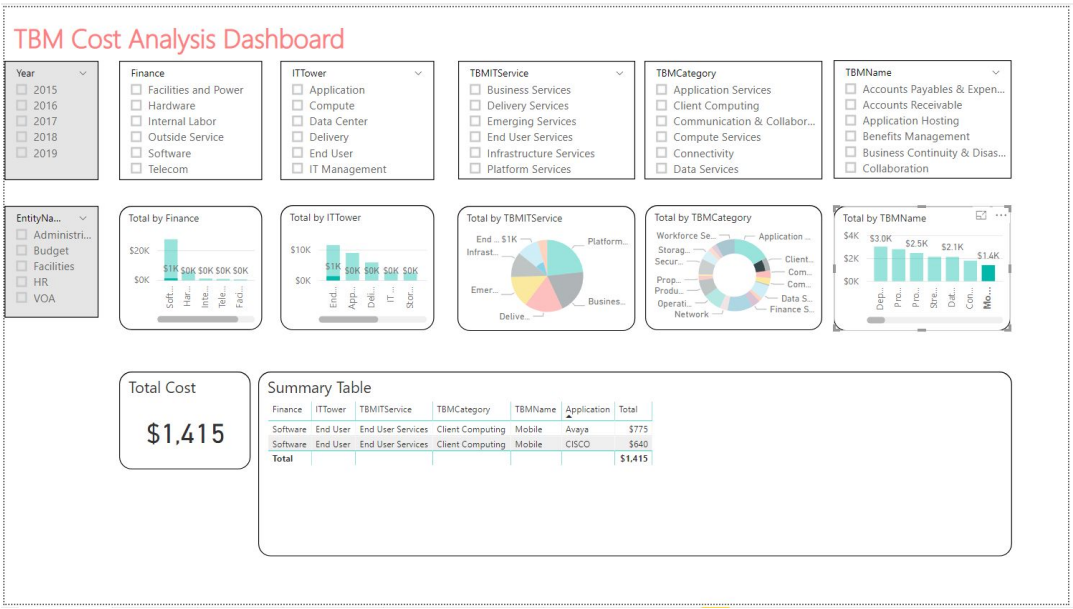


Figure 5.3.2. Cost Analysis Dashboard when selecting ‘Mobile’ category on the last bar graph

For the TBM Cost Analysis dashboard 4.0, only plots of the TBM business layer and its sub-categories remained. All the list slicers were changed to dropbox slicers. The standardized dropboxes were located on the left side of dashboard. Summary tables were removed. All these changes have been made as our client requested.

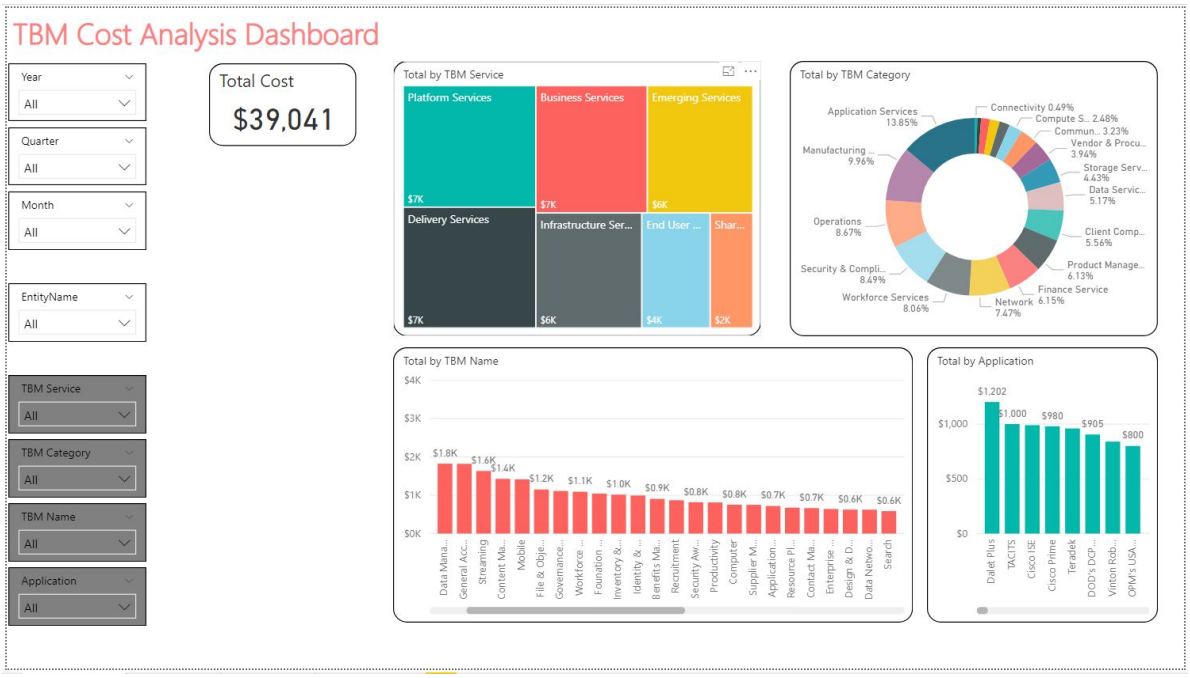


Figure 5.3.3. Cost Analysis Dashboard 4.0

Use Case: Outlier Spending

Suppose the CIO of USAGM wanted to identify any unusual or unexpected spending over the previous five years. He could examine the five year aggregate spending bar chart and notice that application services was the highest TBM Category of spending. It made up 16.94% of all TBM category spending. You can click on 'Application Services' on the bar chart and it will highlight the TBM Names that encompass all application services. Looking at the bar chart, it appears 'Streaming' was the highest TBM name. You can select streaming from the TBM drop down box on the left of the dashboard. You go to application drop down box and select the technologies applications that are using streaming, you see Teradek hardware is costing the most of all the streaming applications. As you investigate, you see there was a hardware refresh that year and you are not alarmed since VOA needs these devices to broadcast their content. You make a note to discuss the next hardware refresh with the VOA director and see if a cheaper option does exist.

When the CIO meets with the director of VOA, they discuss the purchase of Teradek hardware and notice that is was a 'soul source' acquisition, meaning the Teradek was the only hardware considered for purchase. The director and CIO do some research and learn that there is now an alternate product on the market called 'Eurodeck.' They agree that when it comes time for a hardware refresh, they will allow multiple manufacturers to compete against each other, likely lowering future cost.

5.4 Time Series and Entity Analysis

The second and third pages of the visualization are the Time Series Analysis and Entity Analysis dashboards. These two dashboards have an identical formation but different perspectives. For the Time Series Analysis dashboard 3.0, Year, Quarter, and Month slicer were located with a grey background at the very first dashboard. A user could search and analyze total costs by time period using these slicers.

Line charts were interactively changed in terms of year and month when one selected at least one of slicers or plots. The Summary table showed the total costs about TBM names and application names by time period. For the Entity Analysis dashboard 3.0, the names of entities are located in the very first of dashboard with grey background. This dashboard performs in the same way as the Time Series Analysis dashboard does but every result is about total costs of certain entities. The bar chart by entity name and year is used here instead of line charts of the Time Series Analysis

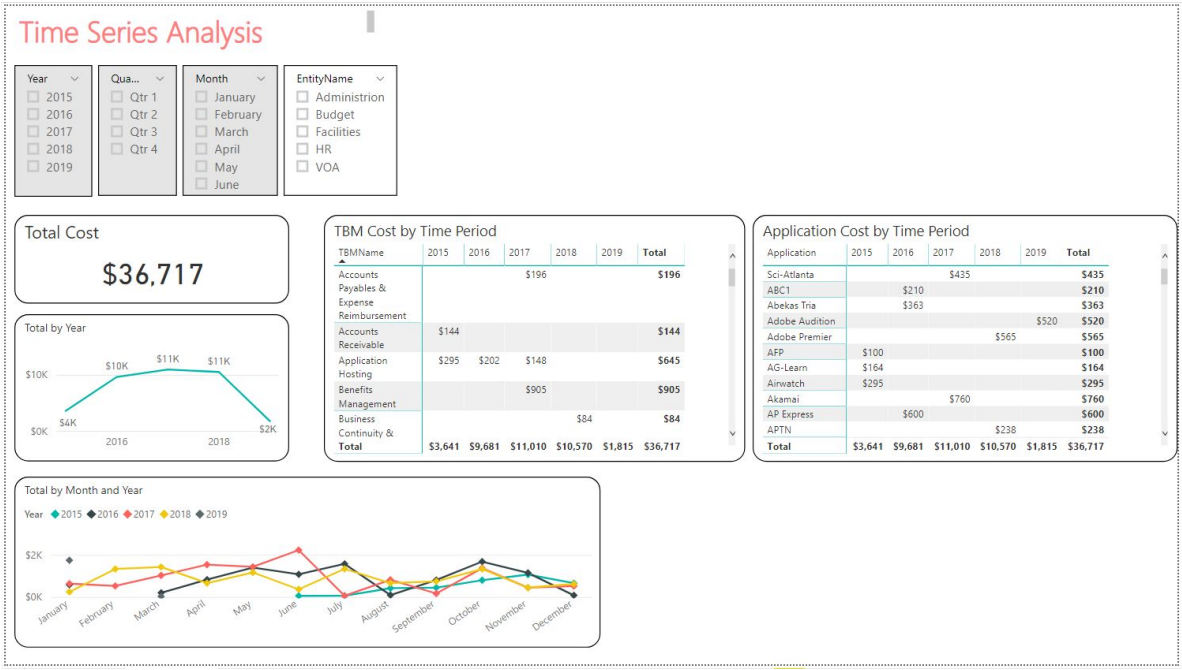


Figure 5.4.1. Time Series Analysis Dashboard 3.0 consists of slicers, graph, card, and table functions

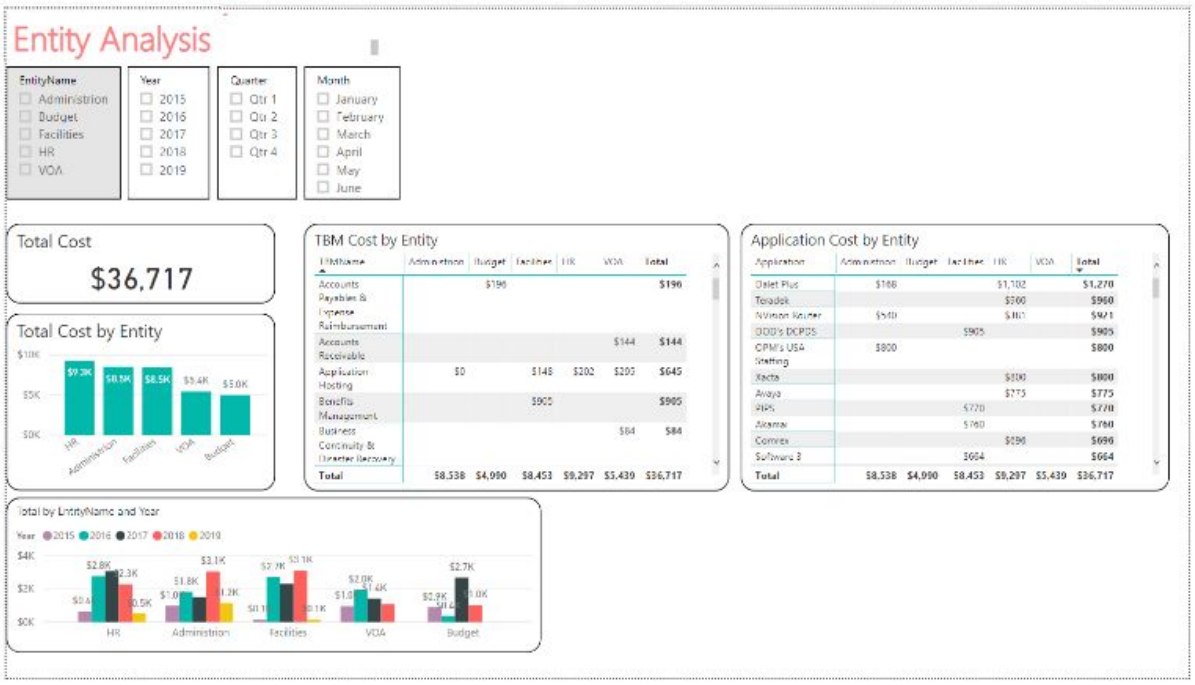


Figure 5.4.2. Entity (Division) Analysis Dashboard 3.0 consists of slicers, graph, card, and table function

Time Series Analysis and Entity Analysis dashboards 4.0 adopted the standardized dropboxes on left side of dashboard like TBM Cost Analysis dashboard. The summary tables were taken out and plots were bigger.

Figure 5.4.3. Time Series Analysis Dashboard 4.0

Use Case: Tracking Expenses Through Time

By monitoring expenses through time, C-Level executives can track if costs are rising or falling and anticipate when technology refreshes are approaching. They are also better able to monitor how well their agency is complying with FITARA regulations. FITARA requires agencies to demonstrate that they are reducing IT expenses over time and finding greater efficiencies in spending.

By examining the Time Series Analysis dashboard, the CFO notices that there was unusually high spending during the month of June 2017. She wants to know why spending has increased so much in that month. Upon drilling down into the data, she sees that there was a hardware refresh for broadcast signal encoding devices. She consults with the CIO and is informed that those were a special purchase for the VOA group. She also learns that this technology is approaching the end of its useful life and a new purchase will have to be made within the next year or two. This information allows USAGM to budget for the new purchase that will be required.

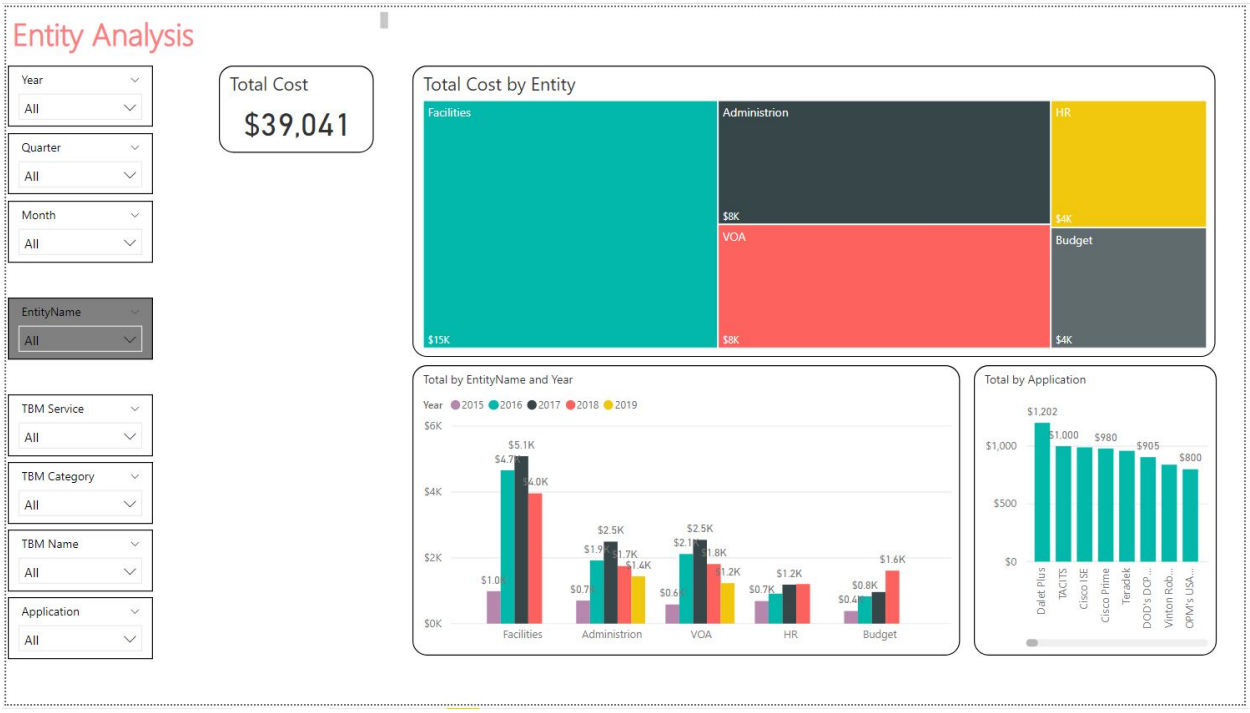


Figure 5.4.4. Entity (Division) Analysis Dashboard 4.0

Use Case: Tracking Expenses by Entity

The CIO can group expenses by entity and track how much money each group has spent year-over-year. Any trends can be identified and information can be learned that might otherwise not have been available. Using this dashboard, the CIO can examine what entities are spending the most and identify any areas of inefficiency or redundancy. In the example Figure 4.3.2 above, facilities has by far the most spend over the last four years. Although HR and Budget has comparatively lower expenses, their cost are rising linearly. This may be an opportunity to investigate to see if there is a variable that is raising the cost. For example, suppose both units own software that is under support but the maintenance fees

are increasing year-over-year. An alternative technology could be identified and procured in the near future to lower overall cost.

5.5 Unit / License Usage Dashboard

Unit / License Usage dashboard has not been change much from version 3.0 to 4.0. Through the Unit/License Usage Status dashboard, any staff in the organization can monitor the number of total units, total seats, used seats, and unused seats of the IT applications and products. Since the dashboard includes a few categorical slicers that can help users to search the units at a certain categorical level, it is convenient to find how many and what kinds of IT application and products are in the same level of categories. The dashboard can function as a base tool to figure out shadow IT because IT applications and products with similar or same functions are shown at a certain level of TBM sub-categories. There is a chance to cut expenditure by removing one of the products.

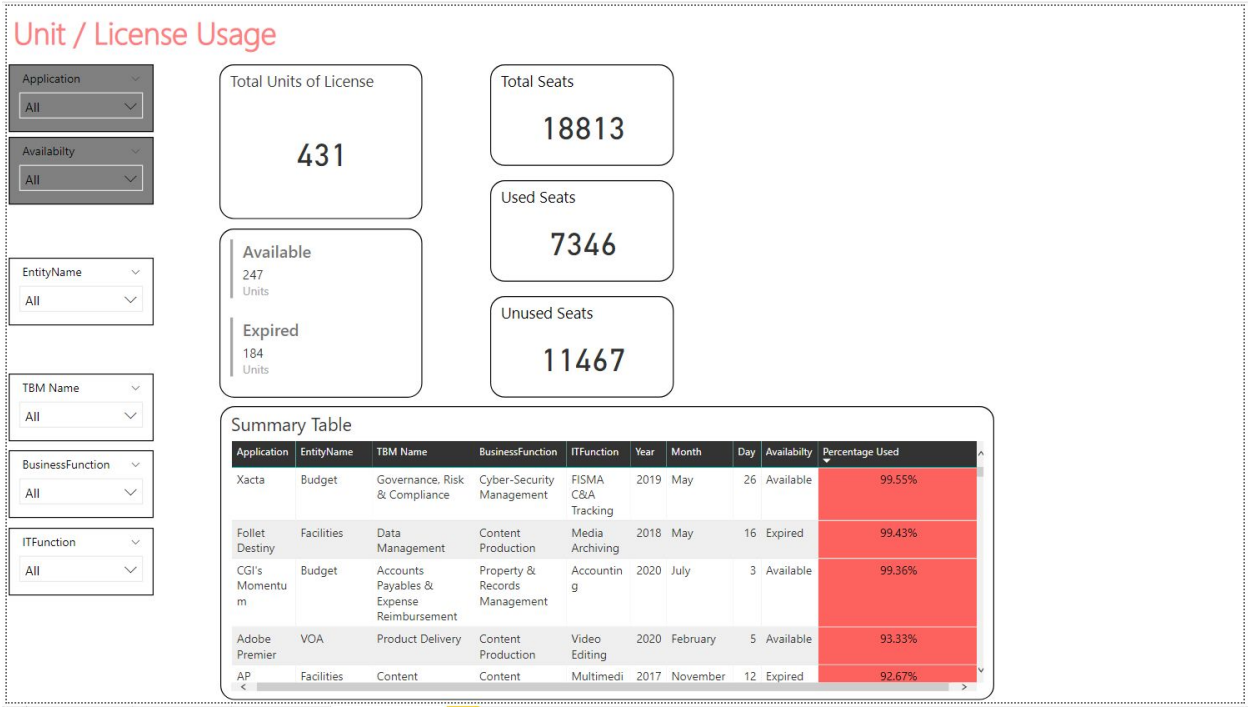


Figure 5.5.1. Unit / License Usage Dashboard 4.0 consists of slicers, graph, card, and table function

Use Case: Unused Licenses

A group of staff members needs more licenses of a statistical programming package like SAS. They intend to submit a purchase requisition to contracting to obtain SAS licenses. Before they do so, they look up the dashboard that tracks license ownership and usage. By using the Power BI dashboard, they can see that the agency already has 115 unused seats for SPSS. They realize they do not need to buy licenses for SAS and instead can immediately utilize the unused seats of SPSS. Not only does the agency save money on not procuring new licenses, they save time since they do not have to wait weeks to months for procurement to buy the new software. They can get started on their project immediately.

5.6 Shadow IT Dashboard

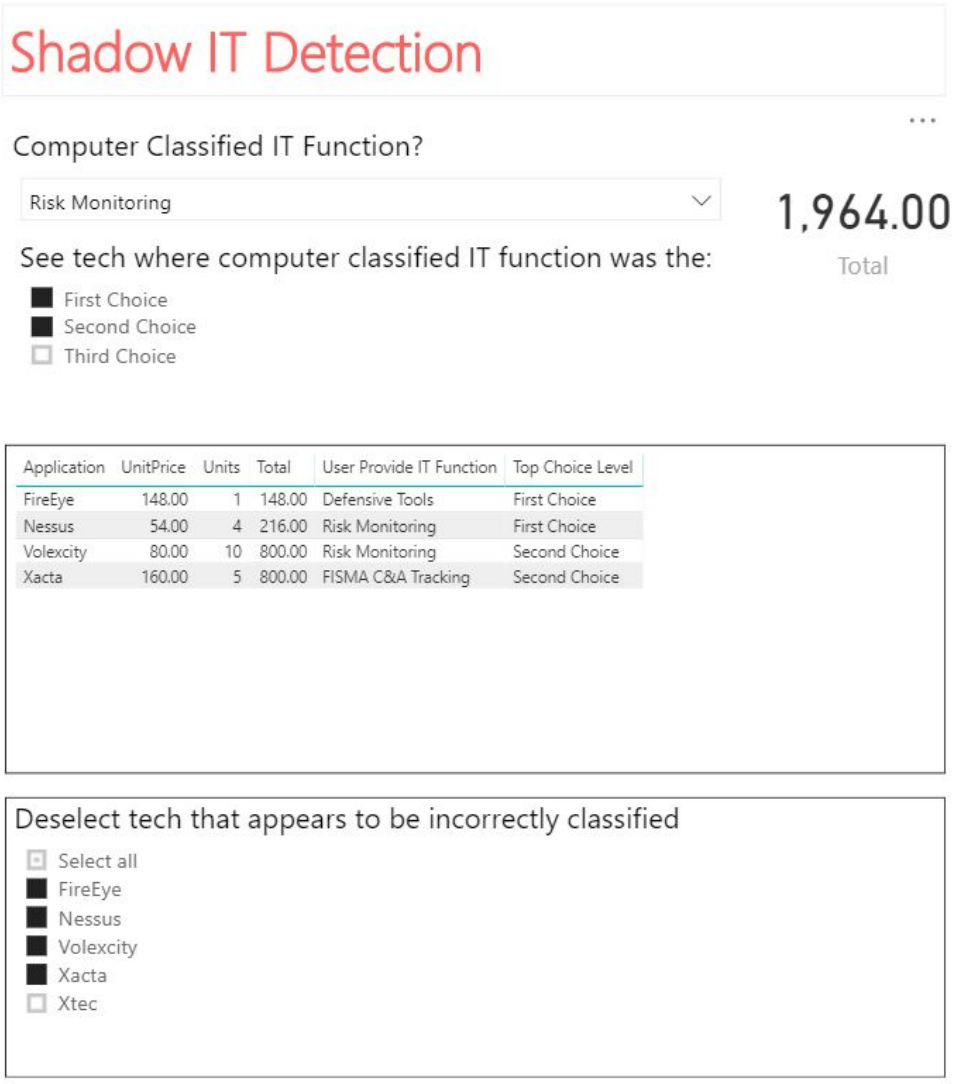


Figure 5.6.1. Our Shadow IT Detection dashboard

The Shadow IT Detection dashboard was added into version 4.0 after the classifier was fully implemented into our web application. The dashboard allows the organization to see technology expenditures by algorithmically classified IT functions. This allows for a more comprehensive look at expenses by IT function since many technologies are associated with multiple IT functions beyond the one chosen by the original submitter.

The top drop-down box allows the user to select the computer classified IT function that the user wants to analyze. The checkboxes, corresponding with first through third choices, allow the user to only see technologies where the computer classified IT function was the classifier's first choice, second choice, or third choice. The table below the checkboxes contains cost-related information about the technologies that have been classified into the chosen computer classified IT functions. The dashboard also contains the ability to remove incorrectly computer classified technologies from the table to more accurately calculate expenditures. The total number reflects the costs of the technologies shown on the table.

Use Case: Products with Multiple Uses

A VOA media group needs a way for reporters, across multiple countries, to collaborate on news stories they are publishing. They would like to edit reports simultaneously and share information as it becomes available. The manager for the group is considering purchasing enterprise licenses of Jive collaboration software. Before he submits the purchase request, he logs onto the website and views the Shadow IT dashboard. He selects 'Collaboration' from the drop down menu to check and see if the agency already owns software that can be used. One of the results displayed is Microsoft Office 365.

A previous user manually categorized Office 365 as word processing software - which it is - but the classifier also identified it as collaboration software. The manager does some investigation and discovers that another group in the agency has unused seats of Office 365. He can now utilize those licenses for his group's needs. Even if the manager needs to buy more Office 365 seats in the future, it will still be cheaper than investing in an enterprise license of a new technology- like Jive Software.

Without the addition of the classifier and accompanied dashboard, the manager would not have known that Office 365 seats were available. Even though they were purchased with the intention of using it for word processing, he can now use these seats for his group's collaboration needs.

6 Summary

USAGM has three primary needs for updating and modernizing its tracking of IT expenditures: identifying areas of spending that were redundant, moving to a centralized storage application, and developing automated visualization dashboarding based on live data. Our solution addressed all three. The classifier groups similar technology into categories and helps users identify hardware and software already owned by the agency that may be underutilized. The database and cloud application incorporates centralized storage and eliminates the need for the MOAS model which was subject to data inconsistency errors. The automated dashboard produces real-time reporting of IT expenditures, eliminating information silos and delivering insight to all levels of the organization- from end users to C-Level executives. When fully adopted by USAGM, this solution will allow the agency to transform their IT Management practice, gain new insights into their needs, and focus on their mission of engaging and connecting persons around the world in support of freedom and democracy.

7 Future Work

7.1 Azure Active Directory

As noted above, one of the hurdles we faced was the lack of administrative powers on our Azure subscription. Without administrative powers, Azure Active Directory couldn't be configured for our web application. Azure Active Directory is important for web application security and the security of sensitive data, such as agency expenditures. As it stands, our workaround for the product is that it is publicly accessible and the Power BI dashboards are also publicly accessible.

Besides security risks, the lack of AAD also means a drop in functionality in Power BI embedding. Originally, we wanted the dashboards to have Create, Read, Update, Delete (CRUD) functionality embedded with them. Without AAD, the dashboards are made publicly accessible and therefore database interaction through Power BI is disallowed. This embedded functionality would be a great tool for projecting What-If? scenarios and visualizing them. *What if we invested over here? Would we still be below budget for 2021?* Visualizing and projecting the What-If? scenarios are not allowed without AAD.

Although we were able to automate some of the dashboard visualizations, we were ultimately unable to automate visualizations based on user input. We had a vision that an administrative users would be able to press a few buttons and move a couple knobs and a dashboard would created automatically based on those inputs. Without AAD, Power BI couldn't be embedded and that communication layer between C# and Power BI couldn't be established.

7.2 Python-to-C# Communication Redesign

Also noted above, one of the more frustrating developments was the communication between our classifier coded in Python and the central web application coded in C#. Although we were able to configure a solution, it's definitely not the cleanest and most efficient. There's duplication of memory usage, speed inefficiency, and extraneous security concerns introduced by our method. One of our ideas that was put on hold due to oncoming deadlines was coding a C/C++ wrapper around the Python code. Due to time constraints and unfamiliarity with C/C++, we decided to go with our current implementation instead.

7.3 TBM Taxonomy

A nice-to-have functionality for the TBM taxonomy and the investment reporting procedure would be for the taxonomy to be auto-loaded to the database. We were unable to create import TBM taxonomy functionality into the application due to time constraints. It was one of our stretch goals that was left unaccomplished.

Appendix A - Technical Content

Classifier Web Application

```
namespace VOAprototype.Classification
{
      public static class Classifier
      {
            private static readonly Uri URI = new
Uri("http://gmubbgpython.azurewebsites.net");


            public static void Init(VOAprototypeContext context)
            {
                  Reset().Wait();
                  var itfunctions = from itf in context.ITFunction
select itf;
                  foreach (ITFunction itfunction in itfunctions)
                  {
                        Add(itfunction.Name,
itfunction.Unigram).Wait();
                  }
                  Train();
            }

            private static async Task<string> GetRequest(string url)
            {
                  HttpClient client = new HttpClient();
                  client.BaseAddress = URI;
                  client.DefaultRequestHeaders.Accept.Clear();
                  client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("text/html"));
                  HttpResponseMessage response = await
client.GetAsync(url);
                  return await response.Content.ReadAsStringAsync();
            }


            private static async Task<string> PostRequest(string url,
string body)
            {
                  HttpClient client = new HttpClient();
                  client.BaseAddress = URI;
                  client.DefaultRequestHeaders.Accept.Clear();
                  client.DefaultRequestHeaders.Accept.Add(new
```

```
MediaTypeWithQualityHeaderValue("text/html"));
                StringContent bodyContent = new StringContent(body);
                HttpResponseMessage response = await
client.PostAsync(url, bodyContent);
                return await response.Content.ReadAsStringAsync();
          }

          public static async Task<string> Filter(string
description)
          {
                return await PostRequest("/filter", description);
          }

          public static async Task<string> Reset()
          {
                Debug.WriteLine("RESET");
                return await GetRequest("/reset");
          }

          public static async Task<string> Add(string name, string
unigram)
          {
                Debug.WriteLine("ADD: " + name + ";");
                string result = null;
                if (name != null && unigram != null)
                {
                      result = await PostRequest("/map/" +
name.Replace(" ", "_").Replace("/", ""), unigram);
                }
                return result;
          }

          public static async Task<string> Wiki(string wikiname)
          {
                string result = null;
                result = await PostRequest("/wikipedia", wikiname);
                return result;
          }

          public static async Task<string> Train()
          {
                Debug.WriteLine("TRAIN");
                return await GetRequest("/train");
          }
```

```
            public static async Task<string> Learn(string name, string
description)
            {
                  string result = null;
                  if (name != null && description != null)
                  {
                        result = await PostRequest("/learn/" +
name.Replace(" ", "_").Replace("/", ""), description);
                  }
                  return result;
            }

            public static async Task<List<string>> Classify(string
application, string description)
            {
                  Debug.WriteLine("CLASSIFY: " + application + "; ");
                  string result = null;
                  if (application != null && description != null)
                  {
                        result = await PostRequest("/classify/" +
application.Replace(" ", "_").Replace("/",""), description);
                  }
                  return result.Replace("[", string.Empty)
                              .Replace("]", string.Empty)
                              .Replace("'", string.Empty)
                              .Split(";")
                              .ToList<string>();
            }
      }
}
```

Classifier Python Application

```
from flask import Flask
from flask import request
import wikipedia
import string
import numpy
import pandas as pd
#import packages related to natural language processing
import nltk
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('perluniprops')
from nltk.tokenize import word_tokenize
```

```
from nltk.corpus import stopwords
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes
from nltk.util import ngrams
from apscheduler.schedulers.background import BackgroundScheduler

app = Flask(__name__)

#test variable to see if the app is still alive
lastcall = 1
stop_words = set(stopwords.words('english'))


# variables stored in memory
scraped = pd.DataFrame()
itfunctions = []
filtereddescs = []
tfidf_vect = TfidfVectorizer(max_features=5000, ngram_range=(1, 1))
tfidf_data = []
Train_X_Tfidf = []
Train_Y = []
OLD_Y = []


def detokenize(tokens):
   return "".join([" "+i if not i.startswith("'") and i not in
string.punctuation else i for i in tokens]).strip()


def filter_sentence(content):
   word_tokens = word_tokenize(content)
   word_tokens= [word.lower() for word in word_tokens if
word.isalpha()]
   filtered_sentence = [w for w in word_tokens if not w in
stop_words]
   filtered_sentence = detokenize(filtered_sentence)
   return filtered_sentence


@app.route("/")
def home():
   global lastcall
   lastcall = lastcall + 1
   return str("Hello World " + str(lastcall))


@app.route("/reset")
def reset():
```

```
    global scraped
    global itfunctions
    global filtereddescs
    global tfidf_vect
    global Train_X_Tfidf
    global Train_Y
    global OLD_Y
    scraped = pd.DataFrame()
    itfunctions = []
    filtereddescs = []
    tfidf_vect = TfidfVectorizer(max_features=5000, ngram_range=(1,
1))
    tfidf_data = []
    Train_X_Tfidf = []
    Train_Y = []
    OLD_Y = []
    return "reset"


@app.route("/wikipedia", methods=["POST"])
def wikithis():
    try:
        wikipage = str(request.data, "utf-8")
        search = wikipedia.page(wikipage)
    except Exception as e:
        return str(e)
    else:
        if search != []:
            return search.content
        return "no wiki"


@app.route("/delete/<itfunc>", methods=["DELETE"])
def delete(itfunc):
    try:
        index = itfunctions.index(itfunc.replace("_", " "))
        del itfunctions[index]
        del filtereddescs[index]
    except Exception as e:
        return str(e)
    else:
        return "Deleted"


@app.route("/filter", methods=["POST"])
def nostop():
    try:
```

```
        description = str(request.data, "utf-8")
    except Exception as e:
        return str(e)
    else:
        return filter_sentence(description)


@app.route("/learn/<itfunc>", methods=["POST"])
def learn(itfunc):
    description = str(request.data, "utf-8")
    global itfunctions
    global filtereddescs
    try:
        index = itfunctions.index(itfunc.replace("_", " "))
    except:
        return "NOTHING TO LEARN"
    else:
        filtereddescs[index] = filtereddescs[index] + " " +
filter_sentence(description)
        return "LEARNED"


@app.route("/map/<itfunc>", methods=["POST"])
def add(itfunc):
    try:
        description = str(request.data, "utf-8")
        global itfunctions
        global filtereddescs
        itfunctions.append(itfunc.replace("_", " "))
        filtereddescs.append(filter_sentence(description))
    except Exception as e:
        return str(e)
    else:
        return "Success"


@app.route("/train")
def train():
    global Train_Y
    global Train_X_Tfidf
    global tfidf_data
    global OLD_Y
    try:
        scrape = pd.Series(filtereddescs)
        scrapeitf = pd.Series(itfunctions)
        tfidf_vect.fit(scrape)
        Train_X, Test_X, Train_Y, Test_Y =
```

```
model_selection.train_test_split(scrape, scrapeitf, test_size=0)
       Encoder = LabelEncoder()
       Train_Y = Encoder.fit_transform(Train_Y)
       OLD_Y = Encoder.inverse_transform(Train_Y)
       tfidf_data = pd.DataFrame({'Train_Y': Train_Y,'OLD_Y': OLD_Y})
       Train_X_Tfidf = tfidf_vect.transform(Train_X)
   except Exception as e:
       return str(e)
   else :
       return "Training"


# this is a keep alive function so that the web app cache never dies
scheduler = BackgroundScheduler()
scheduler.add_job(func=train, trigger="interval", seconds=60*60)
scheduler.start()


@app.route("/classify/<application>", methods=["POST"])
def classify(application):
   try:
       application = application.replace('_', ' ')
       description = str(request.data, "utf-8")
       classified = []
       desc_tokens = filter_sentence(description.lower().replace("_",
" "))
       classified.append(desc_tokens)
       transform_Tfidf = tfidf_vect.transform(classified)
       ## naive bayes
       nb = naive_bayes.MultinomialNB()
       nb.fit(Train_X_Tfidf, Train_Y)
       nb_predict = nb.predict(transform_Tfidf)
       predProbs = pd.DataFrame(nb.predict_proba(transform_Tfidf),
columns=nb.classes_)
       predictions = predProbs.apply(lambda row:
row.sort_values(ascending=False).index, axis=1)
   except Exception as e:
       return str(e)
   else:
       return str(str(OLD_Y[numpy.where(Train_Y==predictions[0][0])])
+ ";" + str(OLD_Y[numpy.where(Train_Y==predictions[0][1])]) + ";" +
str(OLD_Y[numpy.where(Train_Y==predictions[0][2])]))


@app.route("/diagnostics")
def diagnostic():
```

```
   return (str(itfunctions) + "<br><br><br>" + str(Train_Y) +
"<br><br><br>" + str(OLD_Y))
```

Approve Web Application Page (in ITInvestmentsController.cs)

```
public async Task<IActionResult> Approve(int? id)
{
      if (id == null)
      {
            return NotFound();
      }

      var purchaseOrder = await _context.PurchaseOrder.FindAsync(id);
      if (purchaseOrder == null)
      {
            return NotFound();
      }
      if (ViewBag.itfunctionlist == null)
      {
            SelectList selectList = new
SelectList(_context.ITFunction.OrderBy(itf => itf.Name), "Name",
"Name");
            ViewBag.itfunctionlist = selectList;
      }

      List<string> classifiedITFs = await
Classifier.Classify(purchaseOrder.Application,
purchaseOrder.Description);
      purchaseOrder.FirstClassification = classifiedITFs[0];
      purchaseOrder.SecondClassification = classifiedITFs[1];
      purchaseOrder.ThirdClassification = classifiedITFs[2];
      var allPOs = from po in _context.PurchaseOrder select po;
      HashSet<string> firstClass = new HashSet<string>();
      List<string> secondClass = new List<string>();
      List<string> thirdClass = new List<string>();

      foreach (ITInvestment order in allPOs)
      {
            if (order.PurchaseDate.HasValue)
            {
                  if (order.ITFunction.Equals(classifiedITFs[0]))
                  {
                        firstClass.Add(order.Application);
                  }
                  if (order.ITFunction.Equals(classifiedITFs[1]))
                  {
```

```
                    secondClass.Add(order.Application);
                }
                if (order.ITFunction.Equals(classifiedITFs[2]))
                {
                    thirdClass.Add(order.Application);
                }
            }
        }
        firstClass.UnionWith(secondClass);
        firstClass.UnionWith(thirdClass);
        ViewBag.Alternatives = firstClass;
        return View(purchaseOrder);
}
```

ITInvestment Object (ITInvestment.cs)

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace VOAprototype.Models
{
    public class ITInvestment
    {
    [Display(Name = "Id")]
    public int Id { get; set; }

    [Required]
    [Display(Name = "Technology Name")]
    public string Application { get; set; }

    [Display(Name = "Description")]
    public string Description { get; set; }

    [Required]
    [Display(Name = "Entity")]
    public Entities Entity { get; set; }

    [Required]
    [Display(Name = "IT Function")]
    public string ITFunction { get; set; }

    [Display(Name = "IT Tower")]
    public string ITTower { get; set; }

    [Required]
```

```
[Range(1, 10000)]
[Display(Name = "Units")]
public int Units { get; set; } = 0;

[Required]
[Range(0.00, 5000000)]
[Display(Name = "Unit Price")]
[Column(TypeName = "decimal(18,2)")]
public decimal UnitPrice { get; set; } = 0;

[NotMapped]
[Display(Name = "Total Cost")]
public decimal TotalCost
{
      get
      {
            return Units * UnitPrice;
      }
}

[Display(Name = "Seats/License")]
public int? SeatsPerLicense { get; set; } = null;

[Range(1, 1000)]
[Display(Name = "Seats Used")]
public int? SeatsUsed { get; set; } = null;

[Required]
[Display(Name = "Business Function")]
public string BusinessFunction { get; set; }

[Display(Name = "Finance (Cost Pool)")]
public string Finance { get; set; }

[Display(Name = "TBM IT Service")]
public string TBMITService { get; set; }

[Display(Name = "TBM Category")]
public string TBMCategory { get; set; }

[Display(Name = "TBM Name")]
public string TBMName { get; set; }

[Display(Name = "E-GOV BRM")]
public string EgovBRM { get; set; }
```

```
      [Required]
      [DataType(DataType.Date)]
      [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}",
ApplyFormatInEditMode = true)]
      [Display(Name = "Entry Date")]
      public DateTime EntryDate { get; set; }

      [DataType(DataType.Date)]
      [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}",
ApplyFormatInEditMode = true)]
      [Display(Name = "Purchase Date")]
      public DateTime? PurchaseDate { get; set; }

      [DataType(DataType.Date)]
      [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}",
ApplyFormatInEditMode = true)]
      [Display(Name = "Approval Date")]
      public DateTime? ApprovalDate { get; set; }

      [DataType(DataType.Date)]
      [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}",
ApplyFormatInEditMode = true)]
      [Display(Name = "Expiration Date")]
      public DateTime? ExpirationDate { get; set; }

      [Display(Name = "1st Classification")]
      public string FirstClassification { get; set; }

      [Display(Name = "2nd Classification")]
      public string SecondClassification { get; set; }

      [Display(Name = "3rd Classification")]
      public string ThirdClassification { get; set; }
      }
}
```

ITFunction Object (ITFunction.cs)

```
using System.ComponentModel.DataAnnotations;

namespace VOAprototype.Models
{
      public class ITFunction
      {
```

```
        [Display(Name = "Id")]
        public string Id { get; set; }

        [Display(Name = "Name")]
        [Required]
        public string Name { get; set; }

        [Display(Name = "Description")]
        [Required]
        public string Description { get; set; }

        public string Unigram { get; set; }
        }
}
```

Appendix B - Agile Development

Class deliverables such as writing sections of the paper or updating slides are not shown here.

Sprint	**Goals** Epic ● Completed Task ● Unfinished Task (Pushed to following Sprint)
1	Application Research ● Research TBM Taxonomy Familiarize Project Tools ● Power BI Tutorial
2	Application Research ● Investigate TBM report requirements ● Develop IT Solutions Classifier (Proof of Concept) 1 ○ Build wikipedia web scraper ● Develop IT Solutions Classifier (Proof of Concept) 2 (Not fruitful) ● Research sub categories required for dashboard ● Research high level metrics required for dashboard Familiarize Project Tools ● Ingest Excel data into Azure Data Storage ● Connect Azure data into a Power BI dashboard (Proof of concept) ● Power BI Practice (Lynda) ● Web Application document upload (Proof of concept)
3	Application Research ● Implement various classifier techniques and identify best fit ● Build IT Solutions glossary ● Explore Procurement / FPDS site for data gathering ● Incorporate SBA whitelist into IT Classifier Familiarize Project Tools ● Power BI Practice (Lynda) Application Development ● Connect web application to Power BI ● Build mock purchase orders / expenditures document for customer approval ● Connect web application to Python (Proof of Concept) ● Build full data flow prototype of initial product vision
4	Application Research ● Incorporate SBA whitelist into IT Classifier ● Pivot classifier from matching user input to existing descriptions Application Development ● Build Power BI Dashboard 1.0 ● Power BI Dashboard 1.1 (due to mock data adjustments) ● Power BI Dashboard 1.2 (due to feedback)

	• Connect web application to Power BI • Build mock purchase orders / expenditures document for customer approval • Connect web application to Python (Proof of Concept) • Build full data flow prototype of initial product vision
5	Application Research • Audit classification results • Assess scraped data accuracy • Build classifier around mock data • Comment classifier code Application Development • Build mock purchase orders / expenditures document for customer approval • Power BI Dashboard 2.0 (mock data approved with minor changes) • Build full data flow prototype of product vision • Embed Power BI Dashboard 2.0 into website • Connect web application to Python classifier
6*	Uncategorized • Create storyboard from Power BI dashboard data • Comment classifier code Application Research • Build classifier around mock data Application Development • Connect web application to Python classifier • Power BI Dashboard 3.0 (additional visualizations and adjustments based on client feedback) • Refine web application user interface
7**	Stretch Goals • Have classifier learn as items are approved • Improve Python caching and web application object models • Database-stored TBM Taxonomy hierarchy

* Instead of having a 4 week long Sprint 3, we opted to evenly distribute our Sprints and organize into uniform 2 week long sprints so there are 6 sprints

** Sprint 7 is the final week before the paper, poster, and presentation are due. Sprint 7 spans 5 days (5/1 - 5/5).

Appendix C - Common References

AAD	Azure Active Directory
CIO	Chief Information Officer
FITARA	Federal Information Technology Acquisition Reform Act
IT	Information Technology
MOAS	“Mother Of All Spreadsheets” (current solution for VOA)
MVC	Model-View-Controller
OGR	United States House Committee on Oversight and Government Reform
SBA	Small Business Administration
Shadow IT	IT solutions that are used within the organization despite not being CIO-mandated
TBM	Technology Business Management
USAGM	United States Agency for Global Media
VOA	Voice of America