

2015

## Managing Utility Properties: Fire Risk Awareness and Mitigation

Andrew Michael Sanchez  
*University of Redlands*

Follow this and additional works at: [https://inspire.redlands.edu/gis\\_gradproj](https://inspire.redlands.edu/gis_gradproj)



Part of the [Emergency and Disaster Management Commons](#), [Geographic Information Sciences Commons](#), and the [Programming Languages and Compilers Commons](#)

---

### Recommended Citation

Sanchez, A. M. (2015). *Managing Utility Properties: Fire Risk Awareness and Mitigation* (Master's thesis, University of Redlands). Retrieved from [https://inspire.redlands.edu/gis\\_gradproj/248](https://inspire.redlands.edu/gis_gradproj/248)



This work is licensed under a [Creative Commons Attribution 4.0 License](#).

This material may be protected by copyright law (Title 17 U.S. Code).

This Thesis is brought to you for free and open access by the Theses, Dissertations, and Honors Projects at InSPIRe @ Redlands. It has been accepted for inclusion in MS GIS Program Major Individual Projects by an authorized administrator of InSPIRe @ Redlands. For more information, please contact [inspire@redlands.edu](mailto:inspire@redlands.edu).

University of Redlands

**Managing Utility Properties: Fire Risk Awareness and Mitigation**

A Major Individual Project submitted in partial satisfaction of the requirements  
for the degree of Master of Science in Geographic Information Systems

by

Andrew Sanchez

Fang Ren, Ph.D., Committee Chair

Mark Kumler, Ph.D.

August 2015



# **Managing Utility Properties: Fire Risk Awareness and Mitigation**

Copyright © 2015

by

Andrew Michael Sanchez

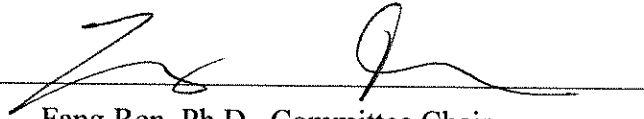


The report of Andrew Sanchez is approved.



---

Mark P. Kumler, Ph.D.



---

Fang Ren, Ph.D., Committee Chair

August 2015



## Acknowledgements

There are a number of people I would like to thank for helping me get through this project and complete the program successfully. First, I would like to thank my advisor, Fang Ren for making sure we examined each little detail to ensure everything worked exactly how it should; despite how dire some of the project stages felt, she was always making sure I knew there was hope for finishing the project. I would also like to thank Giovan Alcala for not only helping me get to the University during MIP season, but also for being there to bounce ideas off of and going through the gauntlet with me, and Neel Chowdhury for helping me break through the worst of my road blocks. When JavaScript seemed hopeless and my models were breaking, he helped me leverage Python to get around these issues and come out with working geoprocessing tools. I also want to give thanks to Nathan Strout, the go-to guy for all things programming in the Center for Spatial Studies. He helped me troubleshoot my Python code, understand WebApp Builder for Developers and the JavaScript behind it, and construct a custom widget to bring my application all together.





# **Abstract**

Managing the Occurrence and Spread of Fire to Transmission Work Sites

by

Andrew Michael Sanchez

Southern California Edison is concerned with fire and the danger it poses to its equipment and employees, and their current strategies are becoming outdated. To help solve this problem, this project developed a web application that allows the user to identify the areas that might be influenced by a fire instance, retrieve information about assets in the danger zone, and display the assets and their risk levels. With this tool, the emergency response teams can more accurately decide what areas are most in need of assistance in a timely fashion. By increasing their efficiency of disaster management, lives and money can be saved.



# Table of Contents

<b>Chapter 1 – Introduction .....</b>	<b>1</b>
1.1 Client.....	1
1.2 Problem Statement.....	1
1.3 Proposed Solution .....	1
1.3.1 Goals and Objectives .....	1
1.3.2 Scope.....	2
1.3.3 Methods.....	2
1.4 Audience .....	3
1.5 Overview of the Rest of this Report .....	3
<b>Chapter 2 – Background and Literature Review .....</b>	<b>5</b>
2.1 Fire Analysis .....	5
2.2 Fire Models .....	5
2.3 Web GIS Development .....	7
2.4 Python .....	7
2.5 Summary .....	8
<b>Chapter 3 – Systems Analysis and Design.....</b>	<b>9</b>
3.1 Problem Statement.....	9
3.2 Requirements Analysis .....	9
3.2.1 Functional Requirements .....	9
3.2.2 Non-Functional Requirements.....	9
3.3 System Design .....	10
3.4 Project Plan .....	10
3.4.1 Requirements Analysis .....	10
3.4.2 Data Acquisition .....	11
3.4.3 Application Development and Testing .....	11
3.4.4 Deployment.....	11
3.5 Summary .....	11
<b>Chapter 4 – Database Design.....</b>	<b>15</b>
4.1 Conceptual Data Model .....	15
4.2 Logical Data Model .....	16
4.3 Data Sources .....	17
4.4 Data Management .....	18
4.5 Data Scrubbing and Loading .....	20
4.6 Summary .....	21
<b>Chapter 5 – Implementation.....</b>	<b>23</b>
5.1 Script Toolset Development .....	23
5.1.1 Asset Selection GP tool .....	23
5.1.2 Asset Selection with Risk GP Tool.....	24
5.1.3 Asset Selection via Upload With and Without Risk GP Tools.....	25
5.2 Web Map.....	26
5.2.1 Publishing Map Service .....	26
5.2.2 Publishing Geoprocessing Services .....	27

5.2.3	Developing Web Application .....	29
5.3	Summary .....	32
<b>Chapter 6 –</b>	<b>Use Cases .....</b>	<b>33</b>
6.1	Asset Selection with Hand Drawn Polygons .....	33
6.2	Asset Selection with Shapefile Uploaded .....	37
6.3	Summary .....	39
<b>Chapter 7 –</b>	<b>Conclusions and Future Work .....</b>	<b>41</b>
<b>Works Cited.....</b>		<b>43</b>
<b>Appendix A.</b>	<b>Feature Set Schema .....</b>	<b>45</b>
<b>Appendix B.</b>	<b>Asset Selection Script .....</b>	<b>47</b>
<b>Appendix C.</b>	<b>Asset Selection with Risk Script.....</b>	<b>51</b>

## Table of Figures

Figure 1.1: Santa Barbara County Study Area.....	2
Figure 3.1: System Architecture .....	10
Figure 4.1: Asset Analysis Conceptual Model .....	15
Figure 4.2: Logical Data Model.....	16
Figure 4.3: Detailed View of Database.....	19
Figure 5.1: Asset Selection Workflow .....	23
Figure 5.2: Asset Selection Interface in ArcMap.....	24
Figure 5.3: Asset Selection with Risk Workflow .....	25
Figure 5.4: Choose to Publish a Service .....	27
Figure 5.5: Service Definition and Analysis Page .....	27
Figure 5.6: Results Tab .....	28
Figure 5.7: Geoprocessing Tool Definitions Page.....	28
Figure 5.8: Web Design Sketch .....	29
Figure 5.9: Web Design .....	31
Figure 6.1: Map View .....	34
Figure 6.2: Select Area of Interest .....	34
Figure 6.3: Assets selected within the area of interest.....	35
Figure 6.4: Asset Selection Output Table .....	36
Figure 6.5: Operational Layer Menu .....	36
Figure 6.6: Asset Selection for Risk Evaluation .....	37
Figure 6.7: Asset Selection with Risk Output Table.....	37
Figure 6.8: Add Shapefile to Application .....	38
Figure 6.9: Shapefile Upload .....	38



## **List of Acronyms and Definitions**

API – Application Programming Interface  
Cal FRAP – California Fire and Resource Assessment Program  
CDF – California Department of Fire and Forestry  
FIM – Facility Inventory Mapping  
GIS – Geographic Information Sciences  
GP – Geoprocessing  
OMS – Outage Management System  
SCE – Southern California Edison  
URL – Uniform Resource Locator





# Chapter 1 – Introduction

Fires are severe potential dangers when working with power grids and electricity. The most effective method of dealing with them can often be simply catching them early and preventing them from spreading. GIS is a useful approach for solving this problem as it provides tools to identify where danger areas are and what level of risk they pose. In this case, identifying important assets and their potential fire risk can make a vital difference in prevention and preparedness.

Southern California Edison (SCE) currently uses the Flex Adobe software kit in conjunction with ArcGIS to evaluate its assets in relation to fire outbreaks. Flex programming is losing support, and with the trend of moving away from Adobe and toward HTML5, SCE needs to update its available applications. This project does just that by creating an application through JavaScript and ArcGIS that identifies assets in a chosen area and returns their type and information as well as their risk for fire.

## 1.1 Client

The Southern California Edison (SCE) Company is the client for this project, with Ms. Erin Garcia as the point of contact. SCE wants GIS used to provide information on its assets in relation to current and future fires. Ms. Garcia provided a cross section of data from their database of assets in order to facilitate the creation of this application and intends to implement it as a replacement for its older system.

## 1.2 Problem Statement

SCE has a fire analysis tool that is outdated and losing support. Adobe FLEX is a programming language designed to allow for consistency across multiple browsers. Esri, however, is reducing support for FLEX based applications in favor of JavaScript. As such, a replacement application is required for SCE in order to properly protect and monitor its assets, particularly where fire is concerned.

## 1.3 Proposed Solution

To address this issue, a GIS application was implemented. This system would examine an area of interest, analyze the asset data and fire risk data provided for given area, and output both a list of contained assets and a map showing the fire risk levels of these assets.

### 1.3.1 Goals and Objectives

The primary goal of this project was to build an application for SCE that helped in disaster management, before and after the fire. This project was aimed specifically at replacing the current methods through which SCE examines and addresses its disaster management, specifically where fire is concerned. With a more modern application, SCE can run through its management tools and have room to adapt them should the need arise.

### 1.3.2 Scope

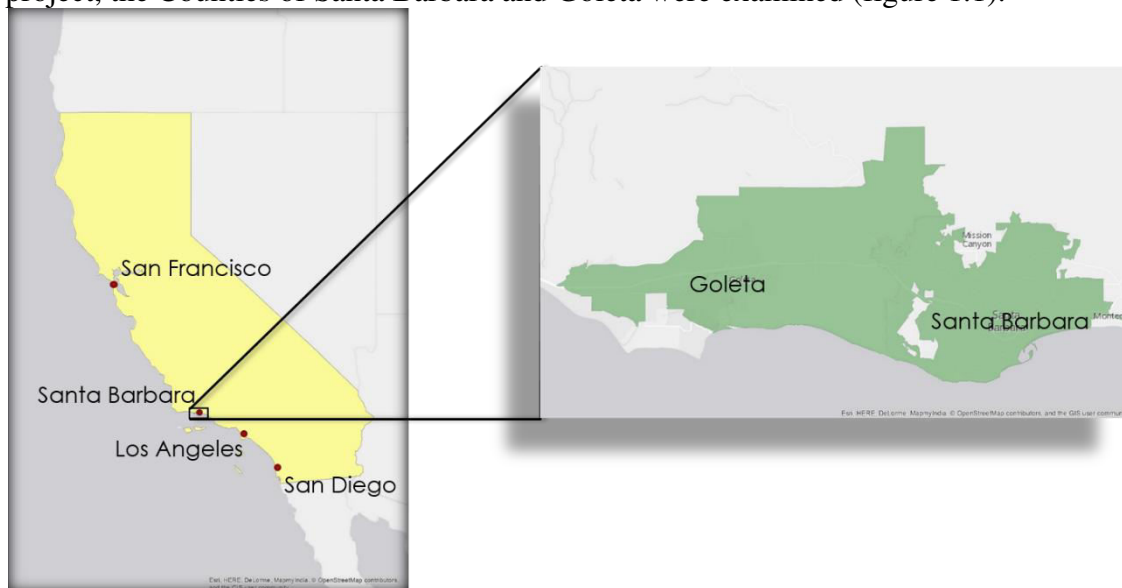
This project was focused on the Santa Barbara area. The data supplied by SCE to create this application are of the city of Santa Barbara, so the data collected for fire risk also fall within the confines of the city. However, the application should be able to handle other areas.

The major deliverable of this project was a web GIS application equipped with five geoprocessing tools all within a JavaScript web application setting. These tools would allow the user to draw a polygon or input the existing shapefile for an area of interest, and then extract assets information within the area and export the information into a table format for further analysis. In addition, the potential fire risks associated with these assets would also be available for user consideration.

Other deliverables included an instructional document containing the processes of the application, as well as training on the implementation of the application.

### 1.3.3 Study Area

In order to test viability and functionality, a study area was chosen for this project. Normally SCE has the entire Southern California Region to monitor, but for the scope of this project, the Counties of Santa Barbara and Goleta were examined (figure 1.1).



**Figure 1.1: Santa Barbara & Goleta Counties Study Area**

### 1.3.4 Methods

The waterfall approach was used for the creation of this project. There were various techniques used throughout the five phases of this project, including requirements analysis, application development and testing, and post development reevaluation. Deciding on the requirements of the project included emails and conference calls with the client, as well as discussions with the advisor on what were acceptable criteria for the application. This ended in requirements including ModelBuilder models that not only selected the attributes, but also evaluated their risk levels. Development and testing of the

applications included use of the JavaScript API for ArcGIS and ModelBuilder from ArcGIS for Desktop. After completion, the application was used to evaluate successful tool completion and reliability of results.

## **1.4 Audience**

This report is geared toward GIS professionals with basic understanding of GIS and ArcGIS and their operations, as well as those from emergency response agencies concerned with spatial risk levels. Contained within will be common GIS terms used to explain the processes behind the application. Therefore, the audience is expected to have the baseline understanding that GIS applications can be created using JavaScript, and how GIS can be used to evaluate and analyze geographic information. This project aims to provide the audience with a way to quickly determine risk for properties of import and thus open up avenues with that knowledge toward preventing disasters.

## **1.5 Overview of the Rest of this Report**

This report contains six other chapters. Chapter 2 gives background information on fire models, JavaScript applications, and a literature review of work on these two subjects. Chapter 3 gives the system design, including the architecture of the application as well as the functions necessary for the application to both work and be acceptable. Chapter 4 houses the database design, giving information on organization, feature classes, relationships and sources for the data. Chapter 5 goes over implementation, how the asset selection tool and fire risk models were added to the application and each step along the process. Chapter 6 is a discussion on the final outputs, what worked, what didn't, and why. Chapter 7 summarizes the results and their implications, as well as introduces potential future work possibilities to further the reach and usefulness of this application.



## Chapter 2 – Background and Literature Review

In order to understand the right approach to this issue, fire management and Web-based applications needed to be explored. Fire analysis uses many different kinds of models. Depending on the area in question, variables that can contribute to fires may increase or decrease in relative value. Understanding which variables are of greatest importance is paramount for creating a tool capable of warning about potential future fires. Python is a powerful object-oriented language that is versatile, including the ability to interact with ArcGIS and its typical suite of tools. This makes it a powerful language for developing geoprocessing tools. Understanding web applications and the languages that go into creating them was important to the functionality of the project.

Section 2.1 includes information on fire analysis and what is necessary for understanding fires. Section 2.2 contains information on fire models and the various forms they can take. Section 2.3 discusses web GIS development and why it is a useful tool to have. Section 2.4 covers the Python programming language and what makes it a good choice for developing geoprocessing tools.

### 2.1 Fire Analysis

Fire spread is a topic that has been increasingly important over the past 40 years. Popular approaches concerning fire management have changed with the last century. Fire fighters and policy writers have had to consider how to take advantage of smaller fires to prevent larger fires (Gollberg, Neuenschwander, & Ryan, 2001). Fire has always been a way nature controls forested areas, manages insects, tree fall-off, diseases, and excess number of trees. The typical approach had been to extinguish fires completely, wherever they occurred, but this caused unintended consequences. Flammable materials built up over long periods. Fire spread then increased, and the resulting damages along with it. Fire spread is a series of individual ignitions dependent on the same conditions: how recent the latest fire was, buildup of fuels, fuel types in the vicinity, ground versus canopy fuel, dry versus moist vegetation, wind, slope, air temperature, release of flammable gases, and aspect. These all have an effect on how fire spreads (Rothermel, 1972). Fuel models vary drastically based on location, but are arguably the most important part of understanding fire before it happens. Fuel models determine the make-up, the potential spread, and the resilience of the fire.

### 2.2 Fire Models

One cannot simply pick any fire model when trying to analyze a place. Different climates drastically affect susceptibility to and likely occurrence of fire. For example, a tropical area contains a significant amount of moisture in the air, which lowers the chance of fire. The bark on the trees is also thick, which further insulates the trees from danger. The plants in the area, however, are not evolved to recover from fire events, so resprouting is difficult, if not impossible. The same goes for an environment like a savanna, where the air is very dry, and the plants are adapted to arid conditions. This makes them fire prone,

but also highly efficient at resprouting, causing damage from a fire to be less extreme (Gomez et al., 2015).

Finding the correct fuel model for an area can be extremely difficult. Fuel density can often change over time in an area, which can make it difficult to predict its potential flammability. Using remote sensing to determine vegetation make-up can also be problematic, as it can be hard to tell the difference between canopy vegetation and ground vegetation (Keane, Burgan, & van Wagtendonk, 2001). All of this comes even before the fire begins. Without this information, all that can be done is suppression around the outside of the fire's reach to try to prevent further spread. Given fuel types and adequate allowance of controlled burning, spread can be better understood and allocation of resources can be better managed. An area that has been burned recently has a shortage of fuel; should a fire reach that area, it will be more easily suppressed, often dying of its own accord, which frees up personnel and equipment to be moved to higher risk locations (Stephens, 1998).

Despite these potential differences between locations and viable fire models, some qualities are shared. Wind direction and speed, slope of location, and azimuthal relationship to north (Finney, 2006); any fire analysis establishes its foundation with these variables. These qualities are important enough that they can be used to create an analysis when some of the more specific and detailed information is not available or too costly to obtain. They are some of the easiest variables to apply constant values to in order to determine an area's general relationship with fire potential. Wind speed and direction tell where a fire will head after it has started. For example, while a field full of dry grass is a fire hazard area, if a fire starts north of it and the area is known for its winds blowing northward, then this area is in less danger than originally expected. This is also why slope and azimuth are important. In the same example, the dry grass field has a slope and azimuth that causes the ground to incline higher, into a large hill further north. This makes it a very high fire risk should the initial fire occur south of the field, but extremely low risk if the fire begins to the north of it. This is because the north facing winds will cause the fire to move north, and if the ground is sloped upwards, it is easier to ignite, as the fire does not have as far to travel. If the fire begins to the north, however, it would need to fight the wind and crest the hill to reach the other side to ignite the fuel. By understanding these variables, important conclusions can be drawn about an area, even without the variables that tend to change frequently, such as weather.

When the goal is management of a current fire, however, the focus changes. While knowing all the aspects of fire behavior is useful in preventing them, these aspects are also just as useful in reacting to them. When a civilian group is endangered by an approaching fire, having a model that demonstrates fire spread can help save lives. (Goldberg, Neuenschwander and Ryan, 2001). Using technology, responders can better analyze fires in real time to determine what neighborhoods, buildings, companies, people, and infrastructure elements are most in danger. This leads to taking appropriate action to protect them. The thirteen fuel models currently available fit various vegetation types and can be adapted for moment-to-moment analysis to further refine fire behavior using real-time data. Building on top of the predictive data already present, first responders have more specific data to direct their options.

## 2.3 Web GIS Development

A useful way to interact with data is through an Application Programming Interface (API). An API gives the ability to program a map for the web. Using a suite of tools designed specifically for each API, a specialized product can be generated for wide consumption. APIs are becoming increasingly important as maps and geographic data are more commonly on the web than on paper. Because of this, it is more important than ever to explore the capabilities of these APIs and understand what they are capable of doing in effectively showing geographic information on the web (Chow, 2008).

JavaScript and HyperText Markup Language (HTML) are the primary languages APIs use to communicate and interact. HTML is used to tell the browser how to display the information. For example, if words or titles should be bold or italic, where they should appear on the page, and font size are all aspects controlled by HTML (Robson, Freeman, 2012). JavaScript is a fully functioning programming language, but despite its name has a very different purpose than the Java programming language. JavaScript is used to manage the Web Browser and the objects and items within it, while Java is used to manage the graphics and networking requirements (Flanagan, 2002). While they are different languages, the fact that they can work together in conjunction with HTML is what gives APIs power to display sometimes complicated geographic information.

APIs, almost regardless of the end-use goal, have a particular neogeographic aim to them: to make whatever process or application being created as simple and easy to use as possible. “Essentially, Neogeography is about people using and creating their own maps, on their own terms and by combining elements of an existing toolset” (Haklay, Singleton, & Parker, 2008, pg 2020). This view on the use of JavaScript is driven toward public use, and is a testament to its versatility as a programming language. JavaScript is like the old adage: easy to use, difficult to master. While there is a wide range of ways to implement JavaScript, it is required only for specified and particular tool creation. The general use tools have been created in such a way to be more user friendly (Flanagan 2002). This provides an open and accessible medium through which to develop maps of varying styles and uses.

## 2.4 Python

When creating scripts to execute automated tool workflows in ArcGIS, the Python programming language is often the method of choice. While Python support seems to come out of the box with ArcGIS, there are important reasons behind this inclusion. Python is an interpreted and modular language. Its interpreted nature means that the code can be run directly from an interpreter such as PyScripter. When declaring variables, the data type does not need to be expressly stated beforehand, and behind the scenes the code does not have to be converted to binary machine language in order to execute (Sanner, 1999).

Python’s modular nature means that at its core, Python is a very small package of information, called a kernel. Modules are then attached to the kernel. These modules contain different functions and tools, allowing any instance of Python application to be highly customizable (Sanner, 1999). The *import* module in particular allows this connectivity. This built-in functionality to the base kernel allows the original Python to search through its library of available modules to connect with based on what is defined



by the user (VanRossum and Drake, 2010). With these considerations, it makes sense that it is the language of choice for ArcGIS. There are modules designed specifically for GIS, most notably ArcPy, which contains all the tools that are found within ArcMap. As a bonus, ArcMap has ArcPy imported into the program structure, so whenever definitions for tools need to be assigned, they can be given through Python instead of visual basic, allowing for greater customization without having to leave the Arc software to create an external application.

## **2.5 Summary**

Understanding what goes into analyzing areas for fire risk helps make the impact of an extreme or moderate threat level meaningful. Being able to allocate resources to a region with a correctly established threat level gives a powerful amount of information to the user. Fires can be anticipated when first responders already know the areas with the highest risks. JavaScript can help deliver that message in a customized medium geared specifically toward geographic education. With dropdowns, displays, and tools all working toward cohesive delivery, the information relayed by these tools can be worked with quickly and efficiently. The next section outlines the requirements the application had to meet, how the system was designed, and what goes on in the background of the application.

## Chapter 3 – Systems Analysis and Design

This chapter discusses the functional and non-functional requirements for the asset detection and analysis application for Southern California Edison. Also covered is the design of the system and why this is the method needed to meet the defined requirements. This chapter also describes the project plan and the expected time of completion in comparison to the actual time of completion for each task within the project.

### 3.1 Problem Statement

SCE has a restrictive and outdated tool for selecting and managing assets. Adobe FLEX is a licensed software as opposed to open source, limiting its capabilities and cooperation with other software. SCE therefore needs a web application that uses JavaScript and HTML 5 in order to increase information extraction and selection functionalities. This project solves these issues through a web GIS developed in WebApp Builder for ArcGIS Online.

### 3.2 Requirements Analysis

The main requirement for the project was a web application capable of examining and evaluating SCE assets, allowing Edison to migrate from Adobe FLEX. A requirements analysis was conducted in order to solidify what this entailed. The requirements are broken down into functional and non-functional requirements.

#### 3.2.1 Functional Requirements

Being able to accurately view data all in one location with degrees of detail varying from use to use was central to the project. Migrating from FLEX to ArcGIS Online means using a different set of tools, and those tools need to be able to deliver useful information to the end user, which means only as much detail as is needed. This project also needed to be a web application. By designing the project with a framework that uses JavaScript and HTML5, it increases the customization options in the future and is an important factor when switching from a licensed software package.

#### 3.2.2 Non-Functional Requirements

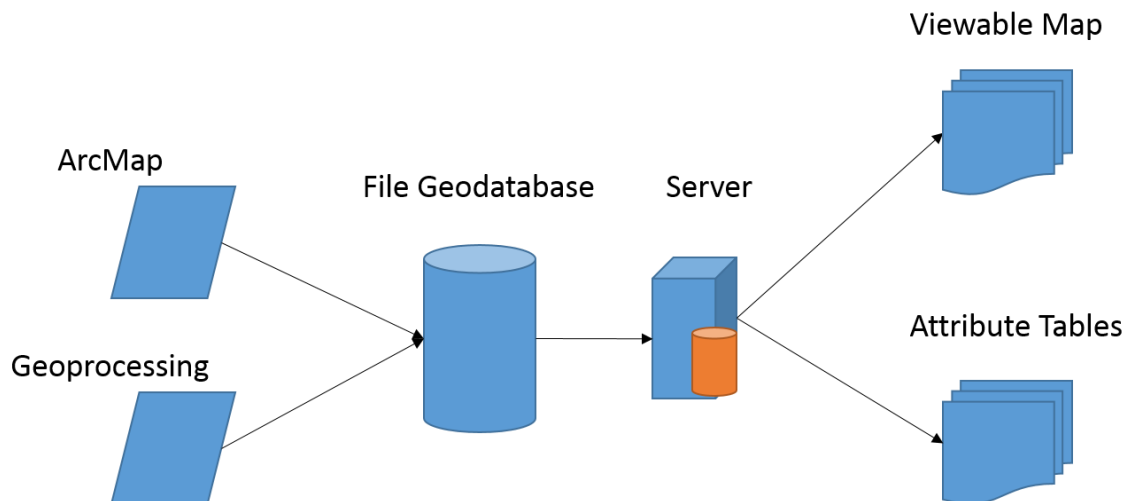
The project began development in a file geodatabase. This means the data could be viewed easily but only edited by one person at a time, should edits be necessary. The purpose of the application is for single instance spatial search and analysis, so the editing restriction was a nonissue. The geodatabase and scripts developed in ArcMap and PyScripter were then published to an online server in order to allow the Web application the ability to refer to and use the published items.

In order to use the application, the user was only required to have access to the internet. The application will not have any security measures to gate access and use; it is for SCE to determine the need for security. Microsoft Excel is also recommended, as the final output tables can be exported to CSV format files for external use.

The transitional requirements for this project included the file geodatabase containing all the data and tables used, the scripts that make up the tools of the application, and implementation documentation. Metadata was added to all data that doesn't already belong to SCE in order to describe need and use.

### 3.3 System Design

The system design is shown in Figure 3.1, which consists of several components.



**Figure 3.1: System Architecture**

The asset and fire risk data is initially managed in ArcMap. Desired data are chosen and imported into an organized geodatabase for storage. The geoprocessing tools are designed in Pyscripter and then linked to a script tool within ArcMap. Once all of these items have been compiled and prepared, they are uploaded to the server as a map service and multiple feature services for web use. With the information now on the web, the data and tools are linked to the ArcGIS Online web application, where the geoprocessing tools are formatted to interact with the data and show the user the assets chosen and the tables containing the attributes for those assets.

### 3.4 Project Plan

This project was developed using the waterfall methodology. The client provided the specifications and desires for the application, but was satisfied with brief conference calls to follow project progression. No prototypes needed to be presented throughout the project lifespan in order to continue progress into later steps.

#### 3.4.1 Requirements Analysis

This task necessitated meetings with the client and with the advisor in order to decide upon the direction and functional requirements of the application. These meetings allowed for mutual understanding of what tools were vital to the customer.

### **3.4.2 Data Acquisition**

This task involved acquiring data on SCE assets from Edison as well as seeking out information on fire risk. Edison provided data on six different asset types (facility inventory mapping (FIM) poles, Substations, outage management system (OMS) Circuits, Sub transmission Lines, Major transmission lines, and work locations). The data on fire risk levels was obtained from the California Department of Forestry and Fire Protection website, [frap.fire.ca.gov](http://frap.fire.ca.gov).

### **3.4.3 Application Development and Testing**

The development of the application took on two phases. The first was the creation of models that explained and executed the workflow of tools necessary to analyze the data. The assets and their various versions (created throughout the execution of the tools) were grouped into datasets based on what stage in the models they were used/created, and the data acquired from Cal Fire were converted from raster form to polygon form for data analysis. The second phase centered on the development of the WebApp Builder application. This medium allows the user to interact with the tools and data. The application was developed in such a way as to allow easy access to the geoprocessing tools and visual representation of the outcome of those tools.

### **3.4.4 Deployment**

The last phase involved handing over the data and scripts developed to SCE. This process included transfer of data, finalizing of documentation, and presenting the application results to an audience.

The major underestimations for this project came in the time required to design and develop the application. Initial versions of the project included JavaScript programming and the Model Builder application of ArcGIS as the core of the application. This was met with numerous road blocks in efficiency and workability, and as such the project had to be reevaluated at an execution level. This forced other objectives, such as project report writing, to be postponed unexpectedly in order for make time to do further development.

## **3.5 Summary**

The main objective of this project was to create an application capable of evaluating data and returning useable and convenient data to the end user. The application had four tools with small but important differences that allowed for specified data retrieval, all within a framework that has more future development than the FLEX suite

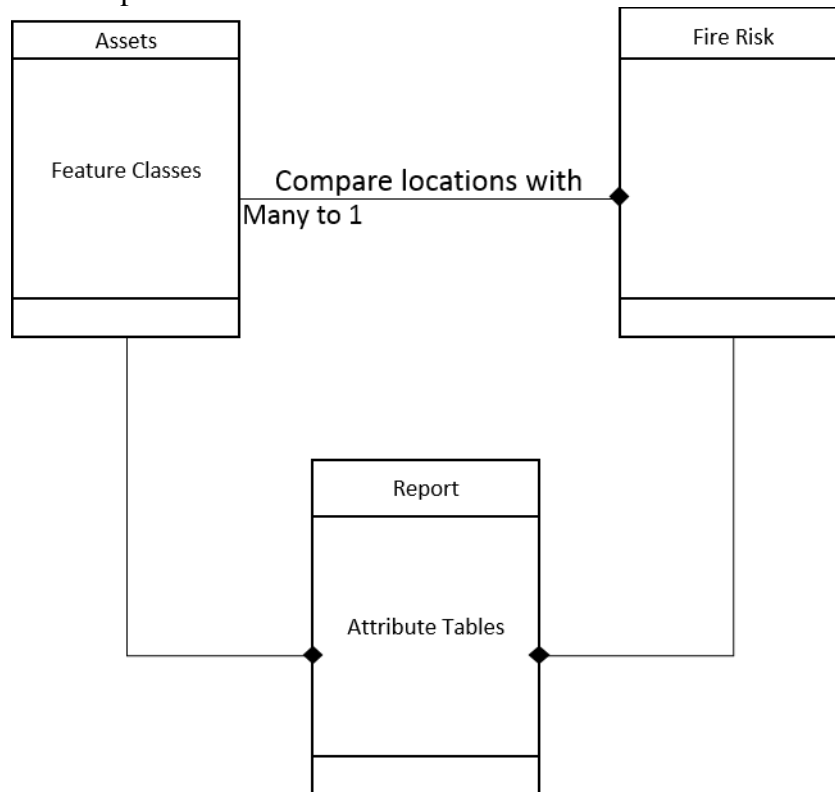


## Chapter 4 – Database Design

This chapter examines the foundations of the project: the conceptual and logical models. The generalized conceptual model explains the high level interactions between the various entities involved in this project. The logical data model details the object types, attributes, and their relationships to each other. This chapter also covers data sources, how they were managed to work for this project, and the data scrubbing and loading procedure.

### 4.1 Conceptual Data Model

The conceptual model is used as an efficient way of expressing the relationships and characteristics of the objects involved in a project. This project concerned the asset information of Southern California Edison in relation to the fire risk levels. As such, three objects were involved in the conceptual data model design. Figure 4.1 shows the overarching relationship between the SCE assets and the fire risk.

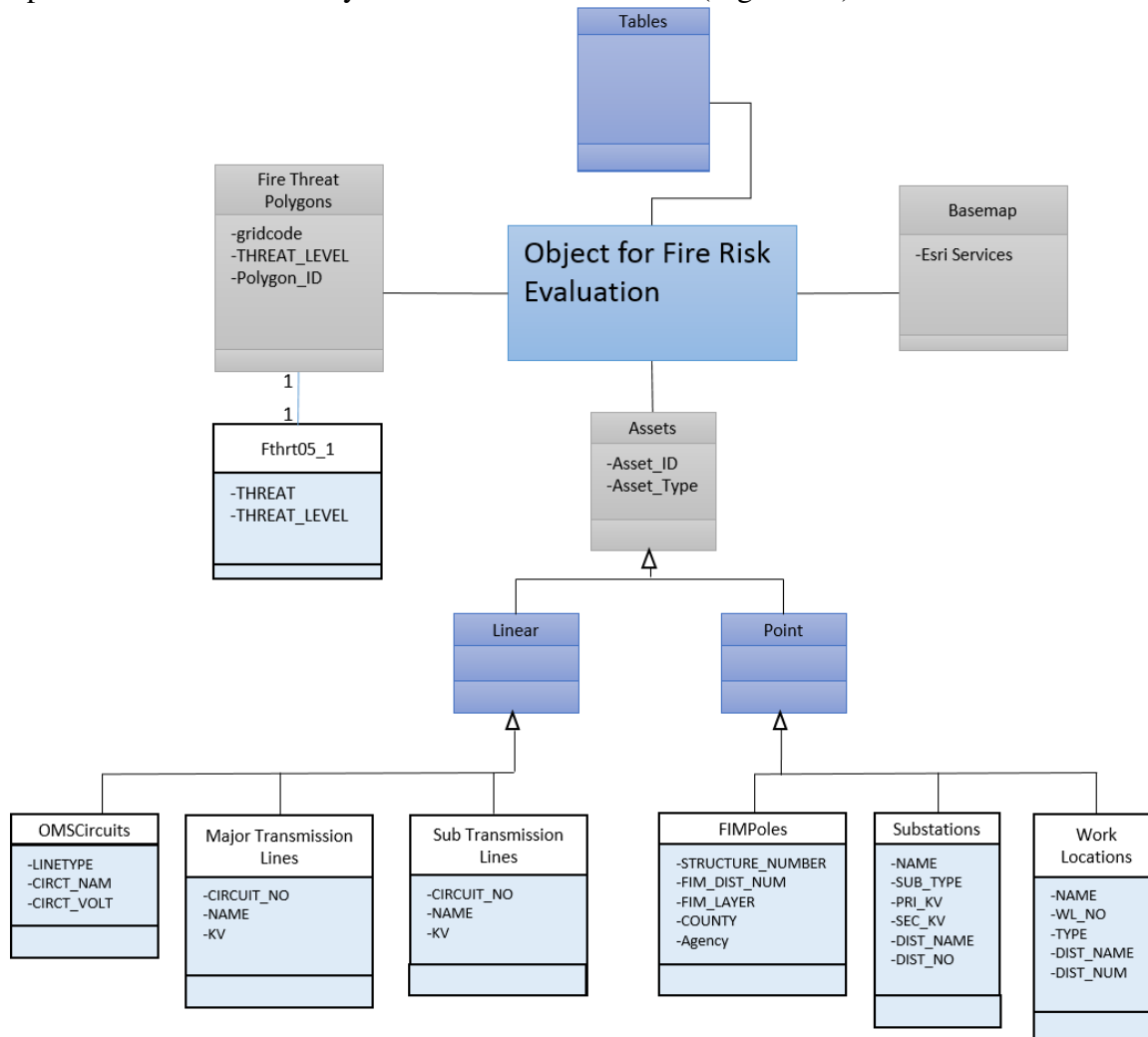


**Figure 4.1: Asset Analysis Conceptual Model**

SCE has tens of thousands of assets in the county of Santa Barbara alone, and being able to associate a base level risk factor for those assets makes planning fire management much easier. These assets take on the fire risk of their surrounding environments. Linking the two together will make it easier to understand which assets might be in danger in a fire emergency. By evaluating fire risk per asset and compiling that information in a report, SCE can make a more informative plan of managing and protecting its facilities.

## 4.2 Logical Data Model

The logical data model helps identify the important attributes for the objects in the conceptual model and how they relate within the database (Figure 4.2).



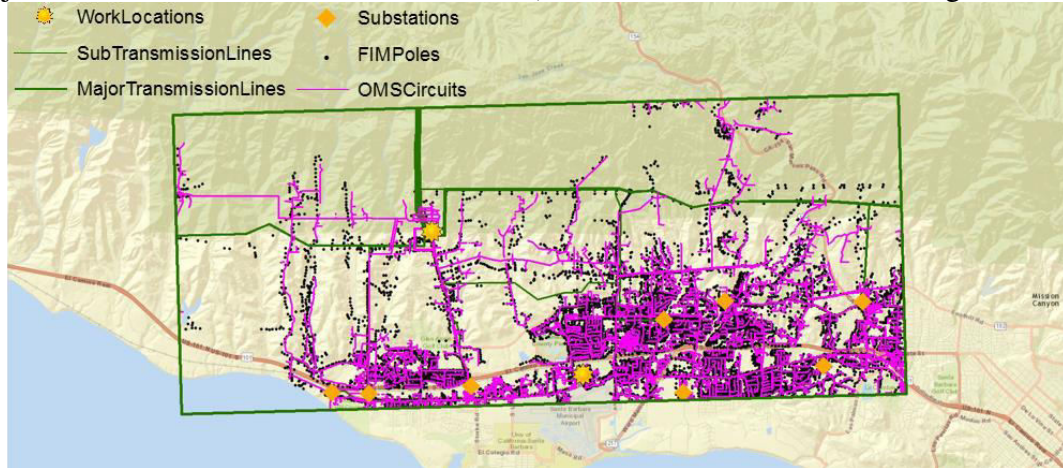
**Figure 4.2: Logical Data Model**

The assets are divided into two kinds by their spatial dimensions, line assets and point assets, and each is further divided into three subtypes of assets. These assets were organized into feature classes in the geodatabase. The line features include sub transmission lines, major transmission lines, and OMS circuits. For the point features, there are facility inventory mapping (FIM) poles, work locations, and substations. Each asset has an Asset\_ID and an Asset\_Type, as well as other associated attributes, to be included in the final report. The report was exported as a tabular dataset. The fire risk level was initially represented by a raster dataset that contains the fire risk information for the state of California. To facilitate the analysis, this raster dataset was converted to a fire risk polygon feature class. It is then associated with the assets based on where they intersect. This allows each asset to receive the fire risk value that is relevant to its surroundings.

### 4.3 Data Sources

The asset data were provided by SCE. The fire risk data were downloaded from the Cal FRAP official website. All data were projected in Web Mercator Auxiliary Sphere in order to adhere to common practices and ensure the widest range of working atmospheres on the web.

Edison provided data on six different asset types (facility inventory mapping (FIM) poles, Substations, outage management system (OMS) Circuits, Sub transmission Lines, Major transmission lines, and work locations). These assets can be seen in Figure 4.3.



**Figure 4.3: SCE Assets**

FIM Poles account for small electrical poles and deployed equipment throughout the county. Substations control electricity power levels, ensuring each building gets the appropriate level of electricity. OMS Circuits help bring the grid back online if there is ever a power outage. Sub transmission lines and major transmission lines are the conduits for transporting electricity from the source throughout the county. Work locations are where SCE personal are physically at, working on fixes or installments in person.

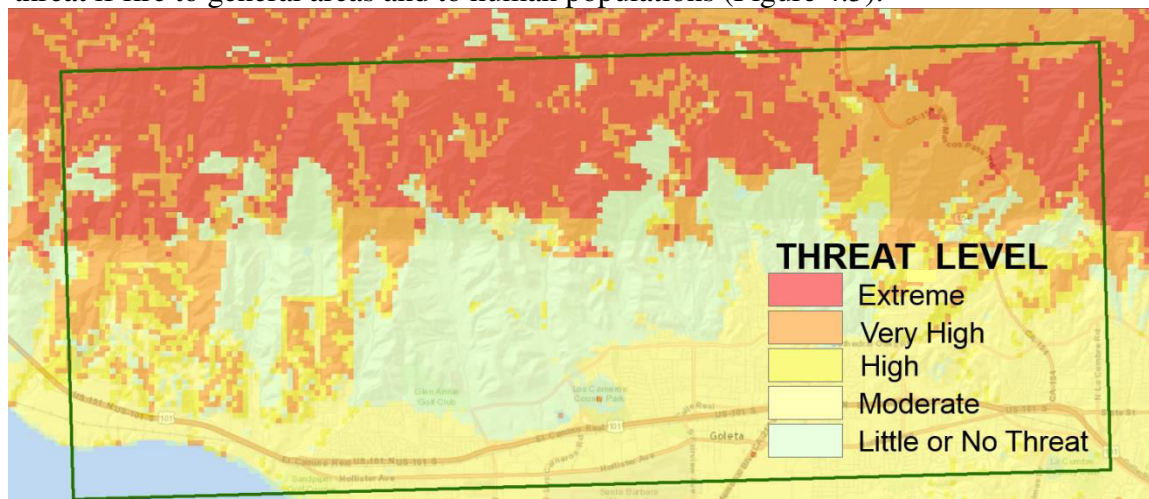
The data for the fire risk levels came in raster format with each cell accounted for in one of ninety-seven attribute rows. Each of these rows accounted for a certain combination of the descriptive fields and represented multiple cells. These rows contain 7 fields (count, fuel rank, frot class, urban, threat, threat 2 people, threat level), and whenever cells match in every category (except count) they are both filed under the same ID and the count goes up (Figure 4.4).



	OBJECTID *	VALUE	COUNT	FUEL_RANK	FROTCLASS	URBAN	THREAT	THREAT2PEOPLE	THREAT_LEVEL
	1	1	740186	1	0	0	1	324	Moderate
	2	2	993424	-1	0	0	-1	324	Little or No Threat
	3	3	3371876	1	1	0	1	324	Moderate
	4	4	2641845	3	1	0	3	324	Very High
	5	5	1060206	3	2	0	3	324	Very High
	6	6	406150	2	0	0	2	324	High
	7	7	4467249	2	2	0	3	324	Very High
	8	8	5360409	2	1	0	2	324	High
	9	9	100642	3	0	0	3	324	Very High
	10	10	312747	-1	1	0	-1	324	Little or No Threat
	11	11	1563430	1	2	0	2	324	High
	12	12	173126	-1	2	0	-1	324	Little or No Threat
	13	13	69614	2	0	1	1	324	Moderate
	14	14	1507	3	1	1	3	324	Very High
	15	15	536	-1	1	1	-1	324	Little or No Threat
	16	16	34603	-1	0	1	-1	324	Little or No Threat
	17	17	2645	1	1	1	1	324	Moderate

**Figure 4.4: Attributes for the Fire Risk Raster**

Threat, threat 2 people, and threat level are all determined by the values present in the fuel rank, frof class and urban fields. Fuel rank is determined by area slope, vegetation and common weather conditions. Frof class is the fire rotation class which looks at fire patterns for each area over the past 300 years and evaluates regularity. Urban is a binary caller which tells whether the cell falls within an urban or rural area. The Department of Forestry and Fire Protection then evaluates these values together to get they threat if fire to general areas and to human populations (Figure 4.5).



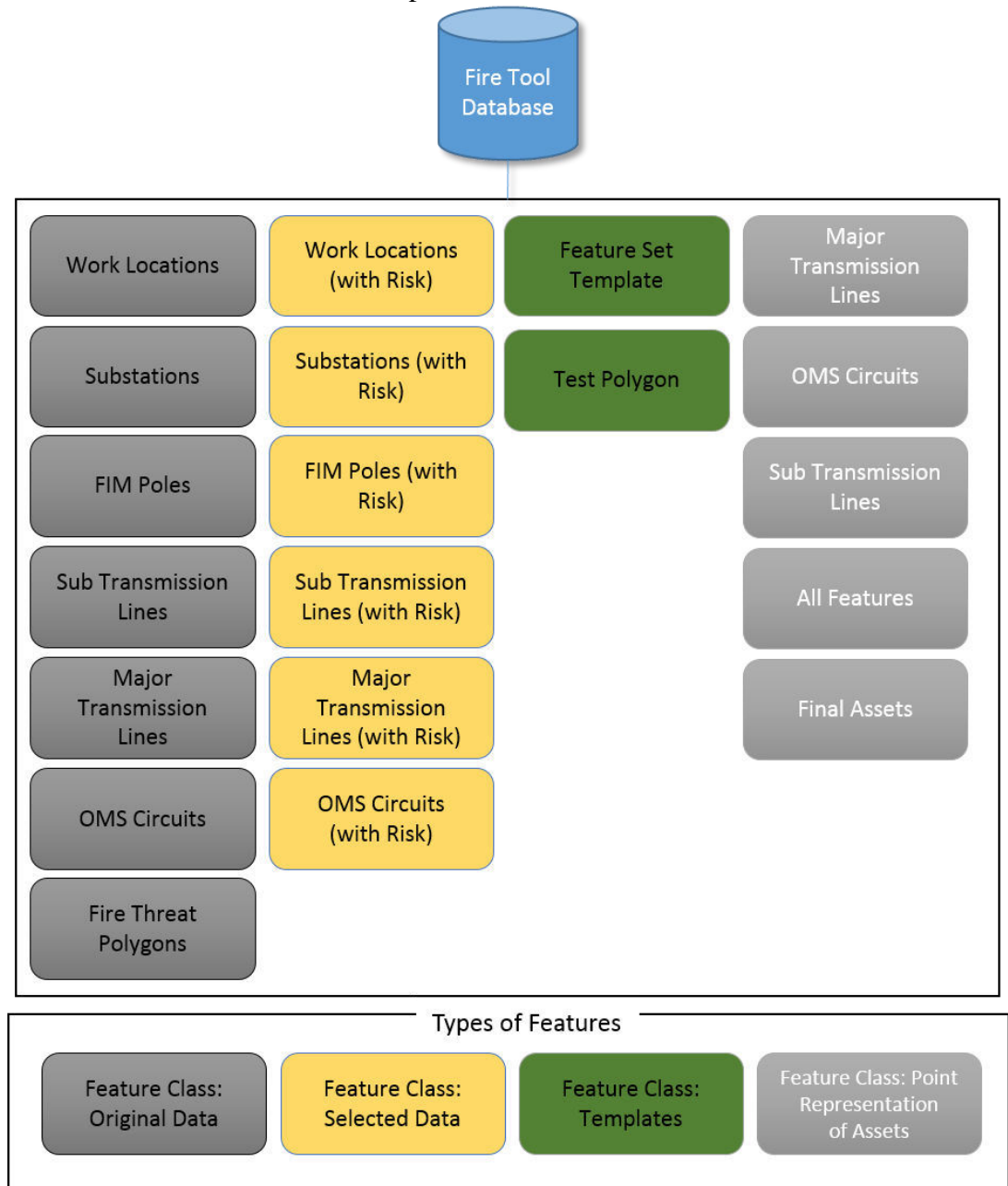
**Figure 4.5: Raster Threat Levels**

For this project, assets are the primary focus. As such, the general areas threat level was chosen over the threat 2 people as the most important threat level for evaluation. There are five threat levels: Extreme, very high, high, moderate, and little or no threat.

## 4.4 Data Management

In order to reduce clutter and organize the project datasets, an Esri file geodatabase was created to house all data. Four feature datasets were created to house the various types of

data: original data, selected data, point representation of assets data, and templates. Figure 4.6 illustrates the database composition.



**Figure 4.6: Detailed View of Database**

The original data consist of seven feature classes. Six of these feature classes are assets from SCE: work locations, substations, FIM poles, sub transmission lines, major transmission lines, and OMS circuits. The seventh feature class is the fire risk polygons needed to find the fire risk levels throughout the search area. In order to get the relationship between assets and the fire risk to work, the fire risk raster was converted into fire risk polygons. In doing so, assets can be linked with the fire polygons of various fire treat levels in a very efficient way.

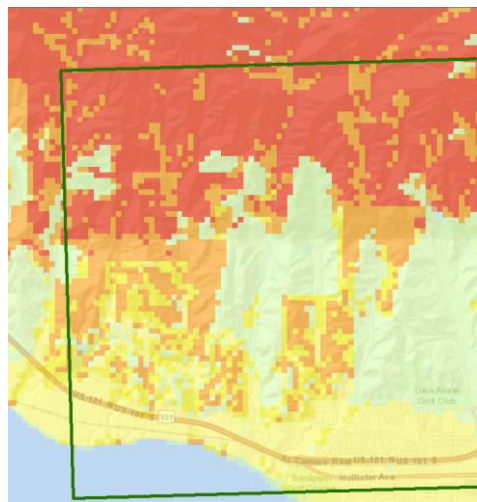
During the analysis, the assets that fall in the area of interest need to be extracted into a single table with their corresponding fire risk levels. To automate this procedure in Python, six empty feature classes will be created for each type of assets to house the selected assets every time the user runs the analysis. Since extracted features of different spatial dimensions cannot be merged into one feature class, a point feature class *Final Assets* was designed to accommodate the attributes of all selected assets. Put differently, every line asset will be represented by a single point with full attributes in the *Final Assets* feature class during the analysis. In addition, a point feature class called *All Features*, which keeps the point representations of all assets, was created to serve two purposes. First, it accommodates the schemas of the six asset feature classes and its schema will be used to prepare *Final Assets* when the analysis is run in Python. Second, this feature class would allow the client to view and export the information of all assets from a single attribute table.

The last dataset houses the templates. This data set contains the feature set template used to define the area of interest as a polygon as well as the test polygon used to test upload functionalities.

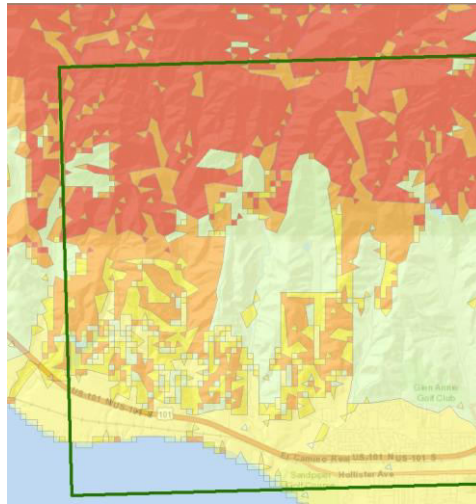
## 4.5 Data Scrubbing and Loading

In order to make the data workable for the Python processes, a few alterations had to be made. The first change was to add two fields to the assets feature classes, *Asset\_ID* and *Asset\_Type*, which would allow the assets to be searched through and actually useable within geoprocessing tools. The *Asset\_ID* was created beginning with a three letter tag representing the asset type and a number to differentiate the assets, such as FIM3565. For *Asset\_Type*, each field was populated with the type of asset it was. Although the attribute value is repeated for all features in the same asset feature class, this information will be conveniently exported to the report.

The second step was conversion. The raster data gathered from Cal FRAP had to be converted into polygons using the Raster to Polygon tool in ArcMap for use in the geoprocessing tools. This change can be seen in figures 4.7 and 4.8.



**Figure 4.7: Raster Form**



**Figure 4.8: Polygon Form**

This change was necessary in order to make the fire risk levels capable of running through the same analytical processes as the asset data.

## **4.6 Summary**

Database design included conceptual models, logical models, data, and data modifications. The conceptual models mapped out the general flow of the application's intent, while the logical model provided information on the actual process the application took to achieve that flow. The data were modified to allow them to interact with the other data.



## Chapter 5 – Implementation

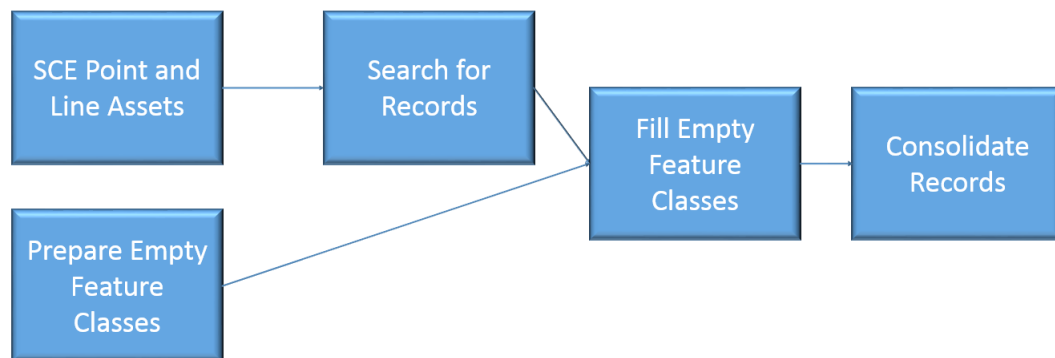
This chapter describes how the web application was implemented. Python scripting, server uploading, tool set up, and web application design are discussed. Script development is found in Section 5.1 and web development is found in Section 5.2.

### 5.1 Script Toolset Development

Southern California Edison (SCE) needed an application that was easy to access in order to manage their assets and protect them from fire. This project created an application through ArcGIS Online. This application contains four geoprocessing (GP) tools, each of which examine SCE's assets and returns information differently. These GP tools were developed using the Python programming language, and will be covered in Sections 5.1.1, 5.1.2, and 5.1.3.

#### 5.1.1 Asset Selection GP tool

The *Asset Selection Geoprocessing Tool* was developed in PyScripter. It allows a user to draw an area of interest and retrieve the records of all the assets within that area. Figure 5.1 illustrates the workflow of this process.



**Figure 5.1: Asset Selection Workflow**

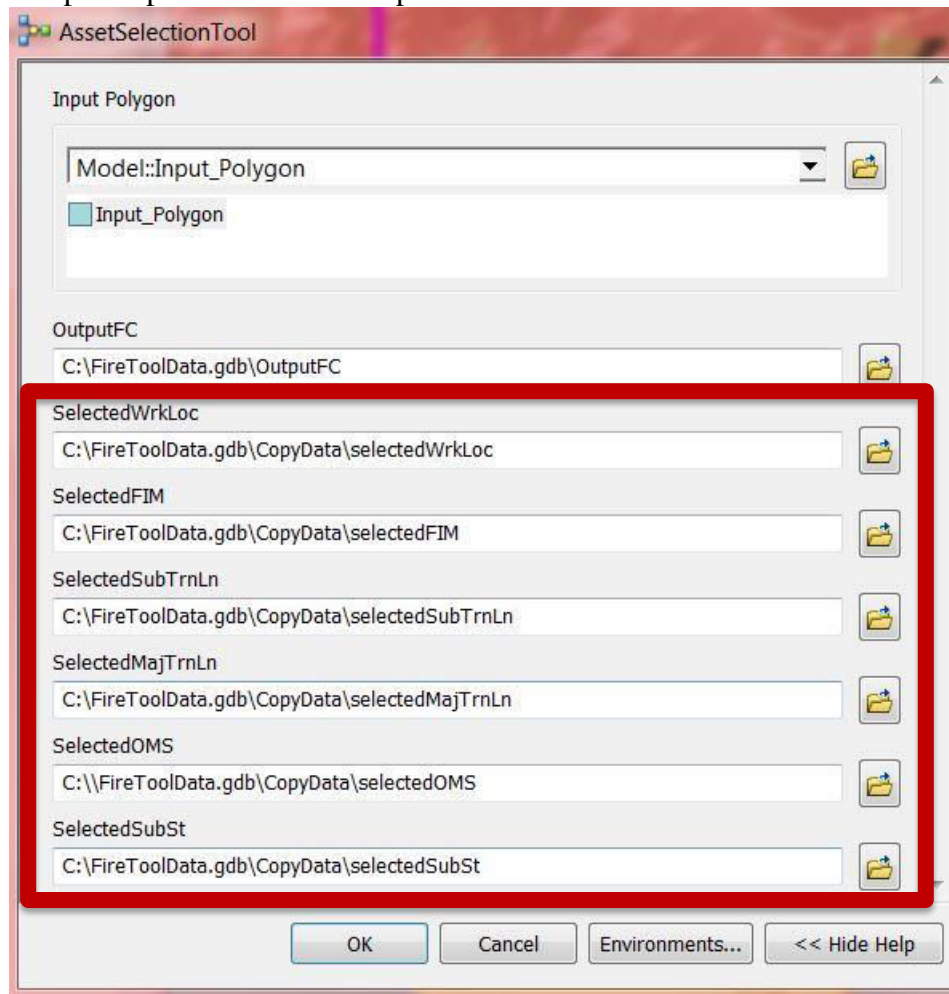
This Python script begins by allowing the user to freehand draw an area of interest. This shape is then applied to the SCE point and line assets. The point assets within the area of interest and the line assets intersecting the area are selected. The script also creates six empty feature classes that maintain the schema of their parent feature classes, the asset feature classes. These feature classes are containers for the selected assets and are stored in the geodatabase. The selected assets are then copied over to the empty feature classes.

For ease of viewing and convenience, the information contained within these feature classes needed to be compiled into one table. To realize this, a new *Final Assets* feature class was created out of the schema of *All Features* (Section 4.3). The selected assets were then appended to the *Final Assets* feature class. The point features were appended directly, while the line features underwent a *Feature to Point* conversion before being



added to *Final Assets*. Although the *Final Assets* feature class does not contain the correct shape of line assets, it was only used to export the attribute information.

Once completed, the script could be run in ArcMap. Figure 5.2 shows the interface for the ArcMap interpretation of the script.

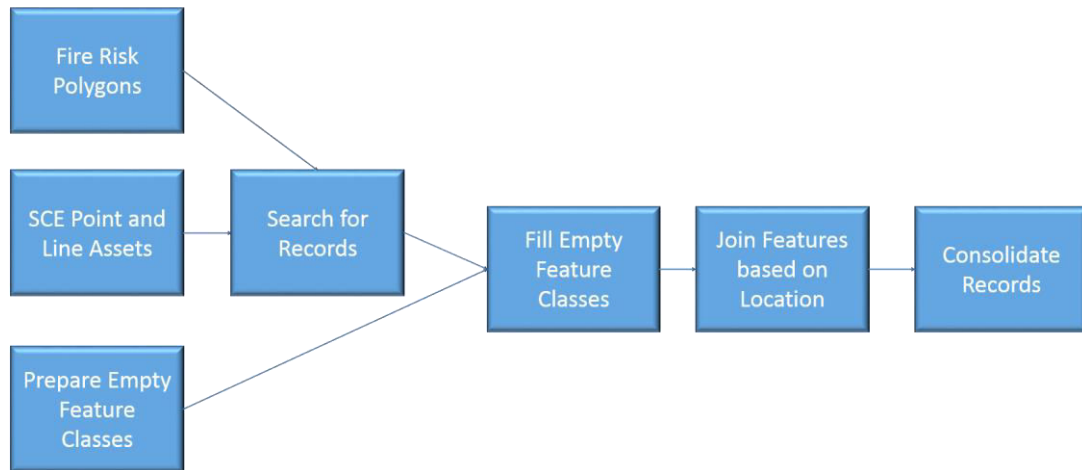


**Figure 5.2: Asset Selection Interface in ArcMap**

While the interface looks complicated, the pathways within the red indicator box are not important for the user. These features are required and created within the processes of the script. The user does not have to define them because the script does that on its own. After the script is published as a geoprocessing service on the web, these fields are not shown, leaving a cleaner, simpler interface for the user to interact with. Despite these extra fields, however, the script executes as expected in ArcMap.

### **5.1.2 Asset Selection with Risk GP Tool**

Compared to the *Asset Selection Geoprocessing Tool*, the *Asset Selection with Risk Geoprocessing Tool* allows the user to select the assets of interest with their corresponding risk levels. The workflow for this tool is shown in Figure 5.3.



**Figure 5.3: Asset Selection with Risk Workflow**

This script begins by allowing the user to freehand draw an area of interest selecting the assets within that area and creating empty feature classes within the geodatabase. After the features are selected, the records are copied over to their empty feature classes for storage. To extract the risk levels of the selected assets, the fire risk polygons are intersected with the selected assets. The field *Threat\_Level* in the fire risk polygon feature class is then added to the newly created feature classes. For point feature classes, this field is added by conducting a *Spatial Join* between the assets and the fire risk polygons. For the line features, however, several geoprocessing procedures are involved. Because a line asset may intersect with multiple fire risk polygons of different fire risk levels, the risk level for the line asset is determined by the dominate risk level associated with the asset. After the intersection, the line asset is broken into segments, each segment receiving the risk level from the corresponding fire risk polygon. Using *Summary Statistics* function on the field of *Shape\_Length* with the case field of *Threat\_Level*, the total length of line segments falling into each fire threat level is summarized. The threat level associated with the longest length is then taken as dominate threat for this line asset. Before the information can be combined, the *Final Assets* feature class is given a new *Threat\_Level* field to house the fire risk data. Finally, the selected assets are appended to the *Final Assets* feature class, now containing their risk levels, in the same export and conversion process as the *Asset Selection Geoprocessing Tool*.

Like the *Asset Selection* script, the *Asset Selection with Risk* script can be run in ArcMap. It has the same interface as its predecessor that becomes more user friendly upon being published as a geoprocessing service for the web. The user only needs to designate the input area of interest and the output location to allow the script to run properly.

### 5.1.3 Asset Selection via Upload With and Without Risk GP Tools

The last two geoprocessing tools are adaptations of the first two. By changing the input from a freehand-drawn polygon to a shapefile, the user can upload predefined areas to examine. The workflows for these two scripts are exactly the same as the *Asset Selection Geoprocessing Tool* and the *Asset Selection with Risk Geoprocessing Tool*, respectively.



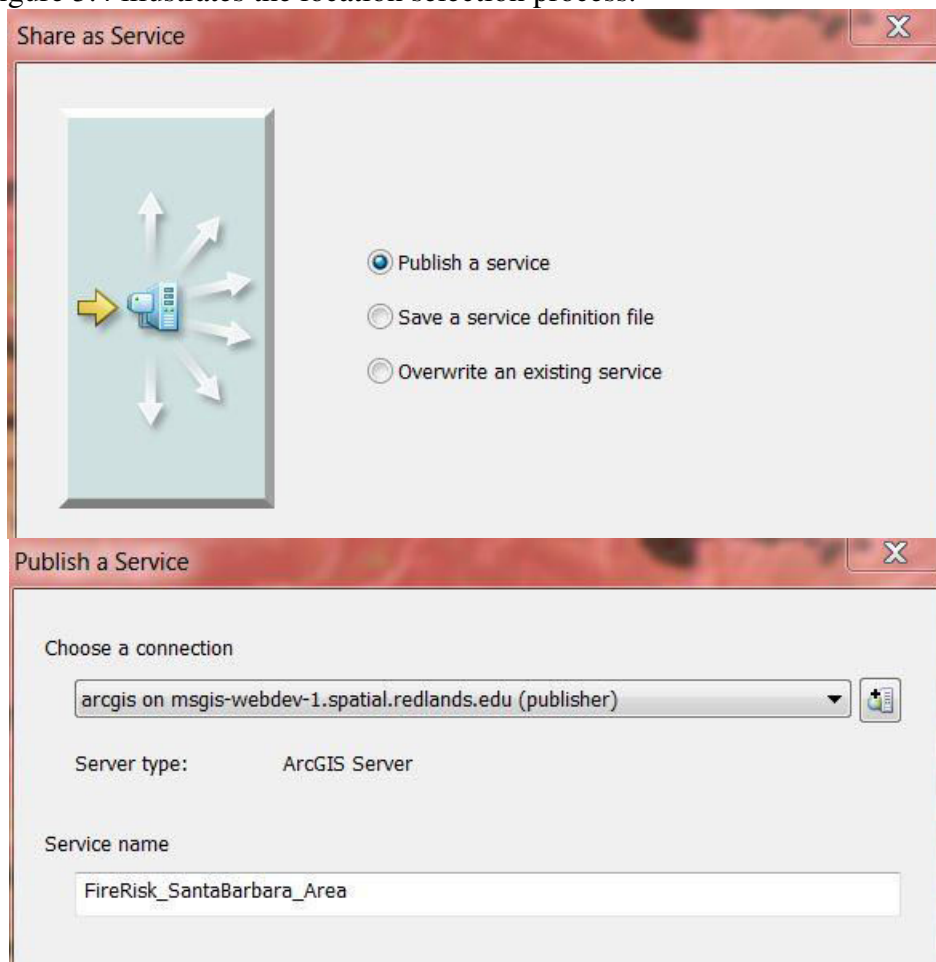
Similarly, these two scripts also run in ArcMap. Instead of drawing the input, however, the user chooses a shapefile as the input layer, either from the catalog window or by browsing the local files. The scripts then execute exactly as before.

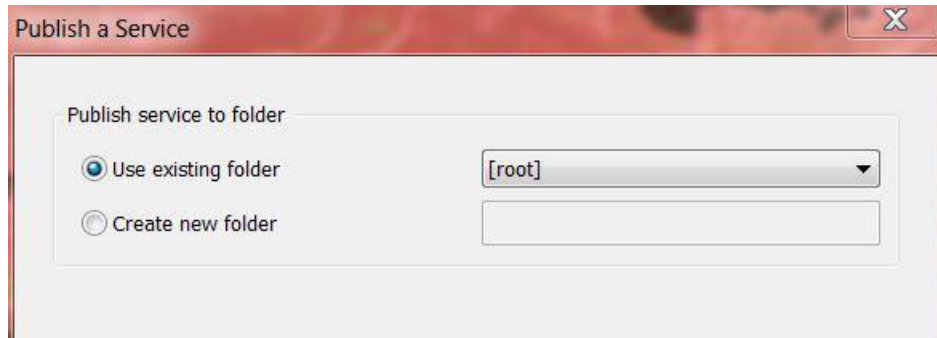
## 5.2 Web Map

Once the four scripts are developed as geoprocessing (GP) tools, they can be published to the web as geoprocessing services. Before this can occur, however, the map document must be published to the web as a map service. These processes begin the conversion from desktop to web. The map service will be covered in Section 5.2.1, the geoprocessing services will be covered in Section 5.2.2. Once these GP tools are published to the web, they can be integrated into a web application, which will be covered in Section 5.2.3.

### 5.2.1 Publishing Map Service

In order to have data for geoprocessing tools to work with, a map service needs to be published. This creates an object on the web similar to a map document (.mxd) within ArcMap for Desktop. To create one, the first step is to designate where the map service is located. Figure 5.4 illustrates the location selection process.

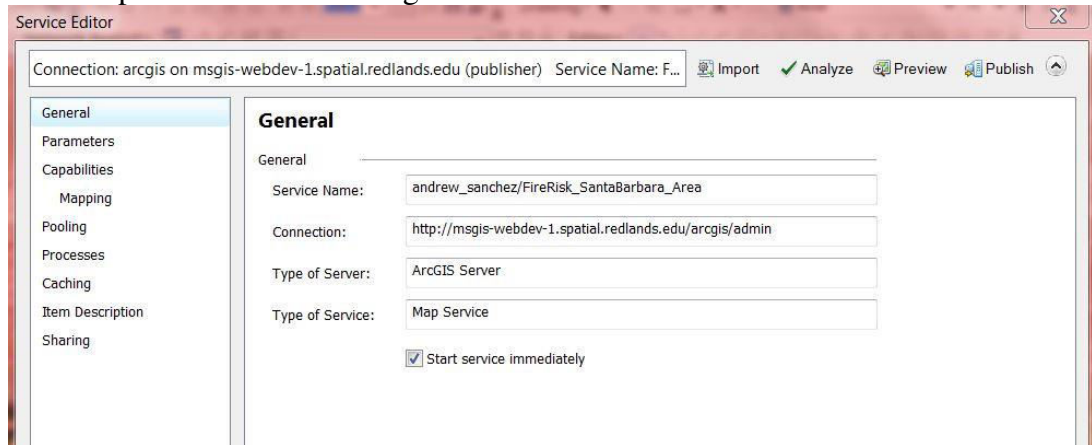




**Figure 5.4: Choose to Publish a Service**

This process is important for both the system and the user. The system maintains the map's existence in this defined area while the user is able to call that location to notify the web application where to look for data.

Once the location has been specified, the proofreading process begins. The main page for this process is shown in Figure 5.5.



**Figure 5.5: Service Definition and Analysis Page**

The definition and analysis of the map service ensures that all the settings are filled in and appropriate for web publishing. Some fields, such as item description, are mandatory in order to maintain metadata and ensure all published maps have a designated purpose. Once all required and desired fields have been filled in, the user analyzes the map service for errors.

Once no errors are found and warnings are either dealt with or ignored, the map service can be published to the web.

## 5.2.2 Publishing Geoprocessing Services

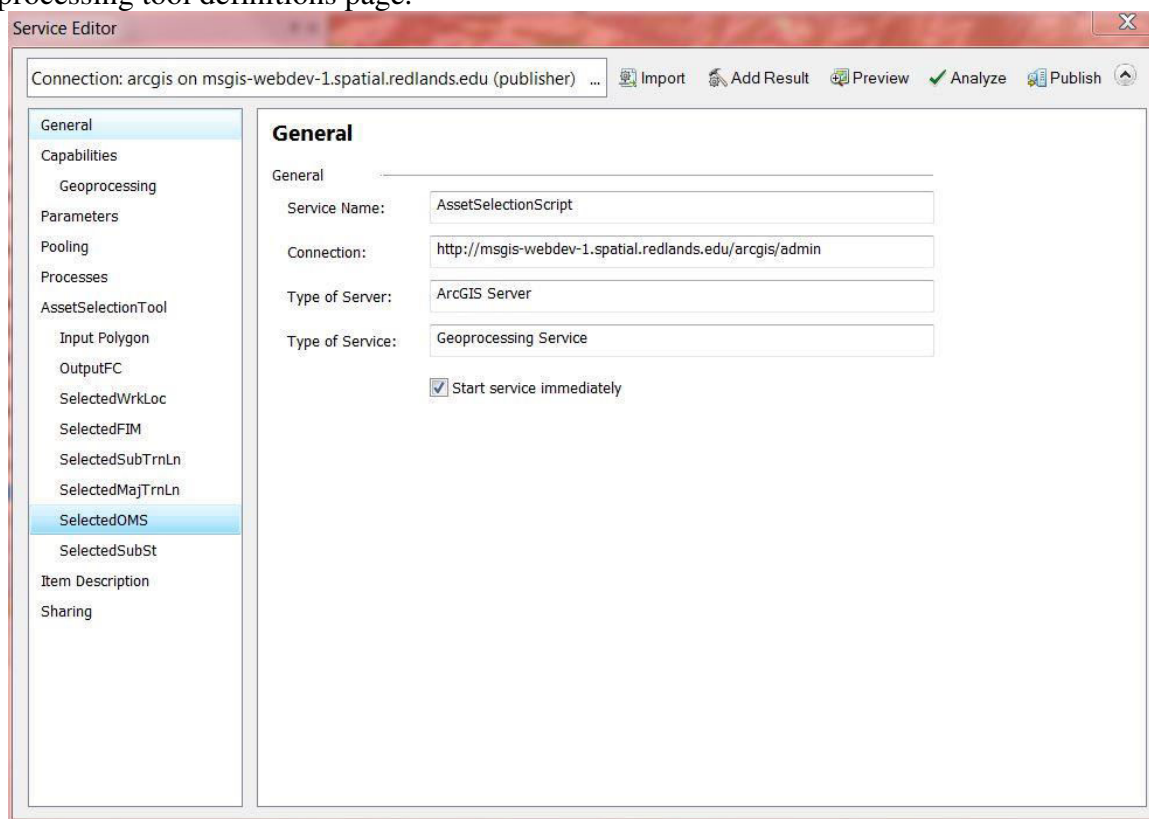
Publishing geoprocessing services happens in much the same way as publishing a map service. To start, the script must be run in ArcMap. Geoprocessing services are published from the *Results* tab. Results can be displayed on screen by going to the Geoprocessing tab at the top of the ArcMap interface and choosing the Results option. This tab is populated with the outcomes of the tools and scripts run within ArcMap. After running

the script, the results tab is opened and the script chosen. Figure 5.6 shows the *Results* tab.



**Figure 5.6: Results Tab**

Similar to publishing a map service (Figure 5.4), the first step to publishing a geoprocessing service is to specify where to publish. Each geoprocessing tool is published one at a time in order to maintain its unique URL. Then, the geoprocessing service goes through its own definition and analysis phase. Figure 5.7 shows the geoprocessing tool definitions page.

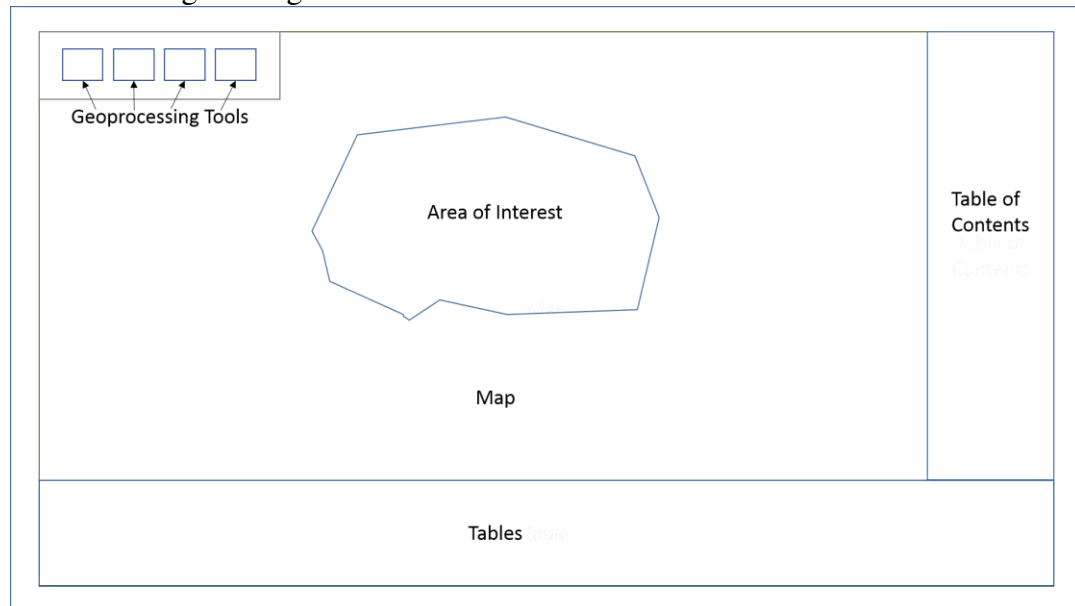


**Figure 5.7: Geoprocessing Tool Definitions Page**

Once the desired and required fields have been filled in, the geoprocessing tool goes through its own analysis process until no errors are found and all warnings are accounted for. This procedure should be repeated for each geoprocessing service. Four geoprocessing tools were published in this way in this project.

### 5.2.3 Developing Web Application

In order to provide the user with an application that is easy and straightforward, a simple interface was designed. Figure 5.8 illustrates the initial sketch of the desired interface.



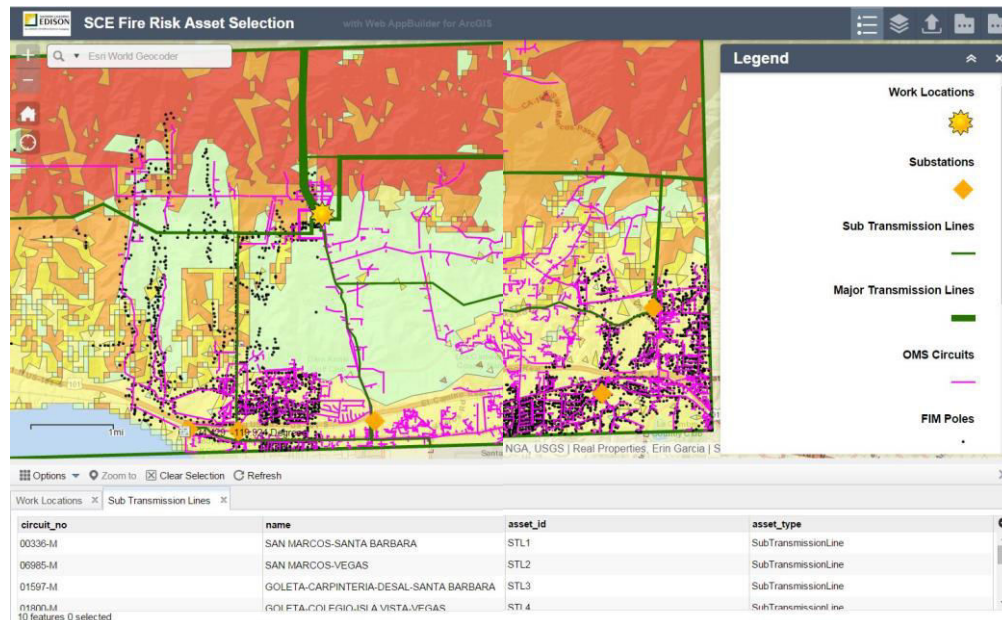
**Figure 5.8: Web Design Sketch**

The goal was to allow the user to easily identify selected assets at a glance. The geoprocessing tools created from the scripts would be at the top left of the screen, the most trafficked portion of a document, given users usually read from top left to bottom right. The map containing the assets and fire risk polygons would populate the center and majority of the screen so that the user can clearly see the region in question. The table of contents, which allows the user to toggle different asset layers on and off, would be located to the right to reduce the clutter with the geoprocessing tools on the left. The tables containing asset attributes would be located at the bottom of the screen, similar to how an attribute table might appear in ArcMap, to create a balanced feel of the interface.

To make the interface function, ArcGIS WebApp Builder was used. This software provides a map interface that has necessities, like a table of contents and a map document, already set up. With the basics predefined, widgets were used to make the application fulfill its unique functionalities. Widgets are stock programs capable of executing various search and analysis functions on the web map. Each one handles a different tool set, and is capable of being customized to fit many kinds of specifications.

Five widgets were added to this application. The first four were geoprocessing widgets, and housed the geoprocessing services published earlier. The fifth widget was a custom widget designed to allow a shapefile representing the area of interest to be uploaded. The ArcGIS WebApp Builder has strict securities on the data allowed in, usually requiring the data to go through a server first. Uploading the shapefile through a server causes complication for the end user. If the shapefile had to go through the server first, the user would have to exit the application and publish the shapefile to their server, a skill the end user may not have. By designing a custom widget, the WebApp Builder allows the shapefile to be uploaded directly to the web application. In order to query the

asset information, the map service that contains the six assets' feature classes was also published to the web application through its custom URL. This populated the table of contents with the various features added to the map as well as the attribute table of the *Final Assets* to display. The detailed code is found in Appendix C. Figure 5.9 shows the conversion from sketch to practical application. The geoprocessing tools live in the top right hand corner, different from their designed location, as it is easily accessible.



**Figure 5.9: Design of the Web Interface**

Each pair of GP tools can be found within a folder with unique names to distinguish them from each other. The zoom in/out options are located in the top left instead of the GP tools. Panning the map is already included in the mouse functionality with the map element by grabbing and dragging. With both the pan and zoom functionalities, the user can quickly center the area of interest and zoom into it. The map is the majority of the page, providing a clear view of the area of the interest. The table of contents is off to the right and the attribute tables are to the bottom. In addition, the user can minimize the table of contents and the table when they are not required on the map to view. This will enlarge the map area facilitating map viewing.

### 5.3 Summary

This chapter covered the development of the scripts and the web application. The Python scripts were created to allow interaction with the asset data. The map and scripts were published to the web, allowing a web map to be created. This web map was turned into a web application so that the scripts could be properly linked with the data to conduct required analyses. This web application was designed to have a simple, straightforward interface to facilitate user interaction and prevent confusion. The next chapter will discuss use cases for this web application.

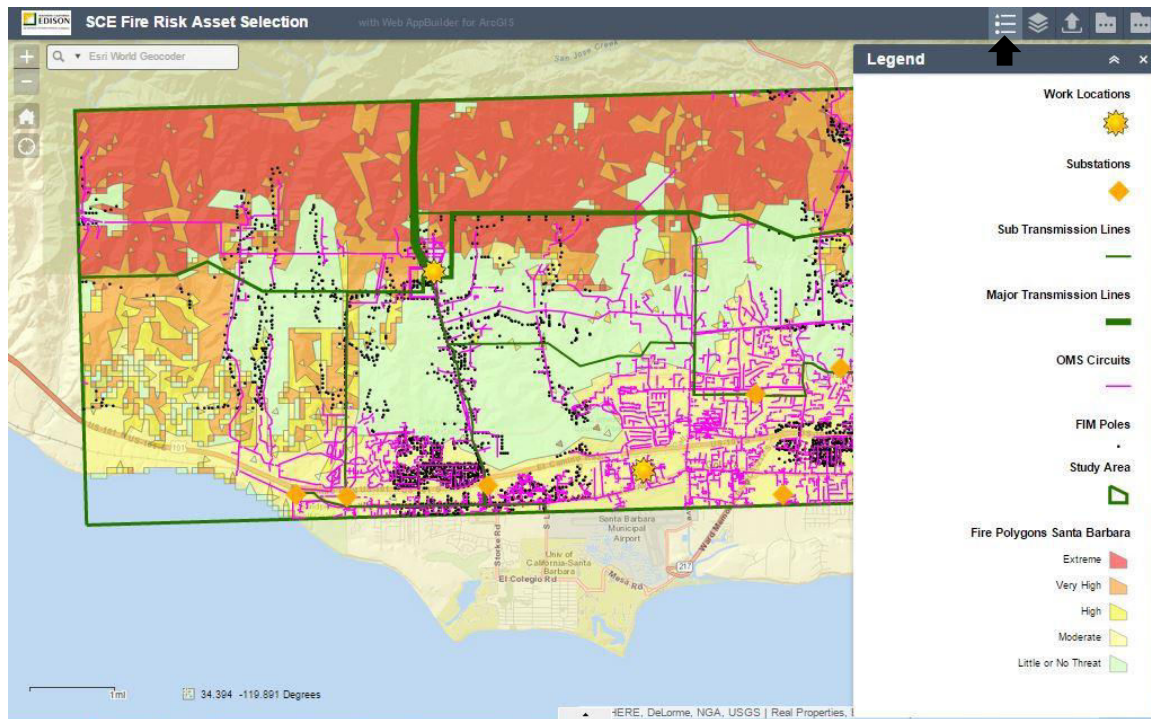
## Chapter 6 – Use Cases

This chapter discusses the use cases for the web application with a focus on the use of the four geoprocessing tools. The geoprocessing tools allow for asset queries at different levels of specificity and detail. According to the client's request, the area of interest would be either drawn on the map or uploaded as a shapefile. The first two tools are executed based on a shape drawn by the user, while the other two tools are run with a polygon shapefile specified by the user. Section 6.1 examines the select assets tools that allow the user to hand draw the area of interest. Section 6.2 focuses on the shapefile upload geoprocessing tools, and the chapter ends with a summary.

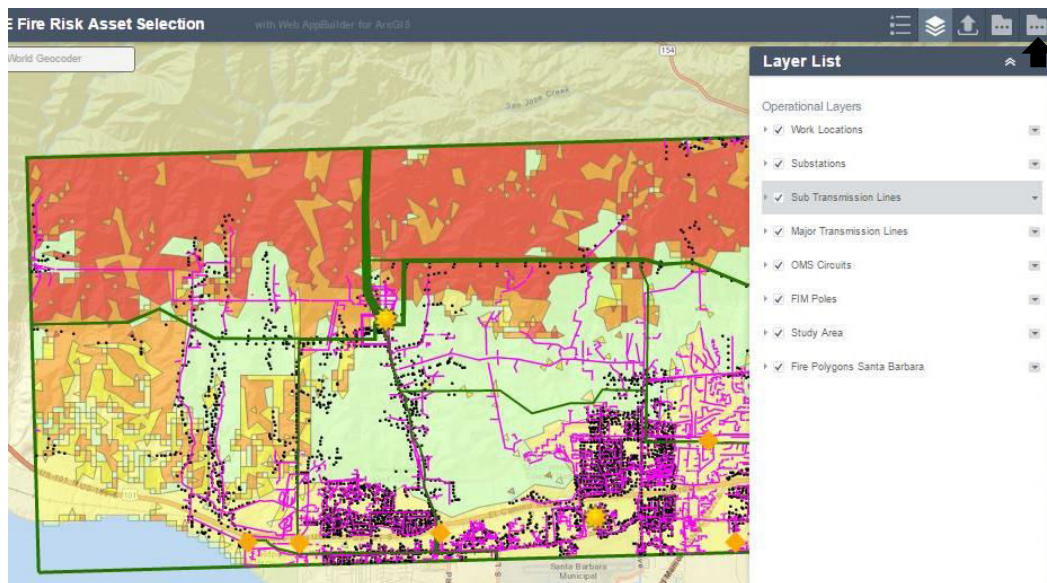
### 6.1 Asset Selection with Hand Drawn Polygons

Two of the four geoprocessing tools work with a hand-drawn area of interest to extract the information about the assets within the area. The difference between these two tools is that one simply returns the assets' information (*Asset Selection*) while the other also outputs the risk levels associated with the selected assets (*Asset Selection with Risk*). Suppose that in a fire instance Southern California Edison's Special Projects Division received an assignment from their supervisor to send information on the assets near the Lake Los Carneros Park in northern Goleta. Figure 6.1 shows the web application interface when the user launches the website and expands the legend. Figure 6.2 illustrates the user changing tabs to view the list of layers. By default, all six types of assets are displayed on the map. The five tabs on the right corner of the window are the information tabs and geoprocessing tools. From left to right the tabs are: legend, operational layers, add shapefile widget, geoprocessing tools for drawing, and geoprocessing tools for uploading shapefile. To better view the map, the user can choose to toggle the layers on and off and read the legend.



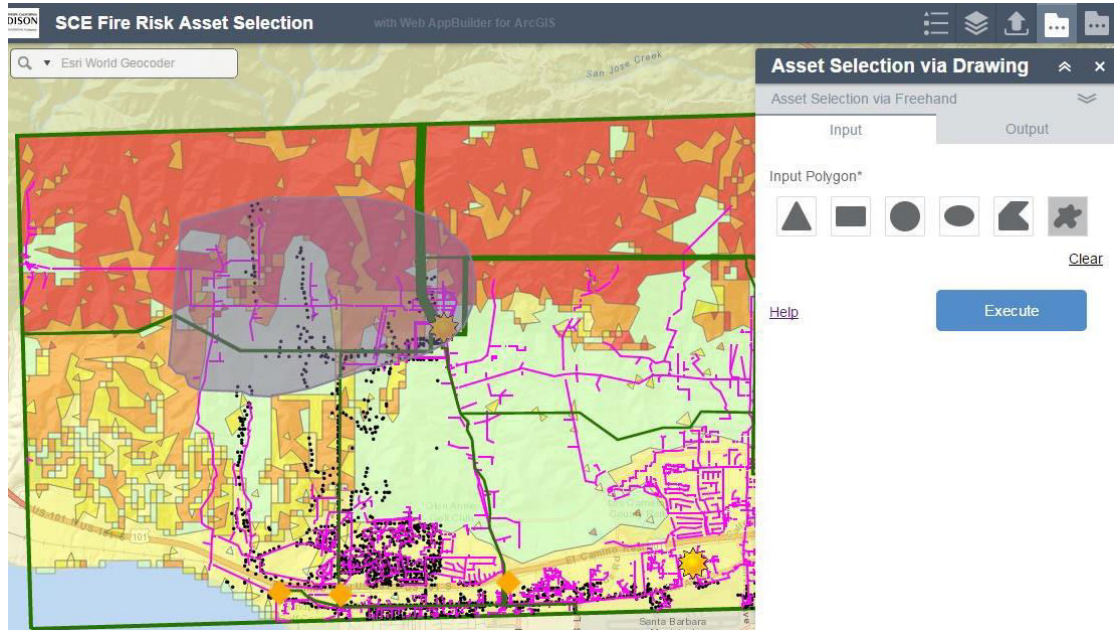


**Figure 6.1: Legend Overlay**



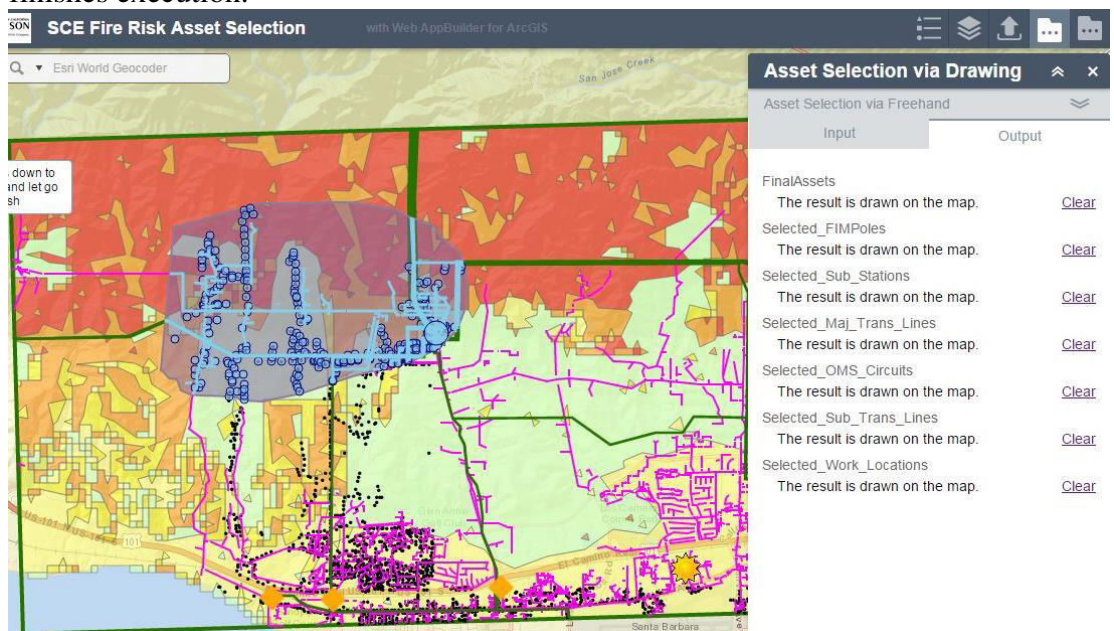
**Figure 6.2: Layer Overlay**

After zooming to the fire instance area, the user opens the *Asset Selection* tool by clicking the *Asset Selection via Draw* tab (Figure 6.3). A variety of shapes are available for users to choose to draw the desired area of interest.



**Figure 6.3: Select Area of Interest**

Once the user draws the area of interest as shown in Figure 6.3, he/she clicks the *Execute* button to conduct the query. Figure 6.4 shows what the user would see after the tool finishes execution.



**Figure 6.4: Assets selected within the area of interest**

The dialog box changes from Input to Output where there are seven different outputs. The outputs consist of the feature classes that only contain the selected information. Six are the individual feature classes for each asset type, while *Final Assets* contains all the information from those six feature classes. The highlighted points and lines on the map represent the point and line assets that fall within or intersect with the



area of interest. From here, the user can examine the tables for each of the feature classes individually or review all the selected assets in a single combined table under the map. Figure 6.5 shows the *Final Assets* table displayed. The other six tables contain the selected assets of the six types respectively and are useful if the user is interested in only particular kind(s) of assets. These tables can be opened one at a time as the user requires by navigating to the operational layers tab.

FinalAssets x						
OBJECTID	CIRCUIT_NO	NAME	KV	ORIG_FID	Asset_ID	Asset_Type
		GAVIOTA				
317	00330-M	GOLETA-SAN MARCOS	2	5	STL7	SubTransmission
316	00324-M	GOLETA-VEGAS	2	4	STL6	SubTransmission
315	00339-M	GOLETA-SANTA BARBARA	1	3	STL5	SubTransmission
314	01800-M	GOLETA-COLEGIO-ISLA VISTA-VEGAS	1	2	STL4	SubTransmission

**Figure 6.5: Asset Selection Output Table**

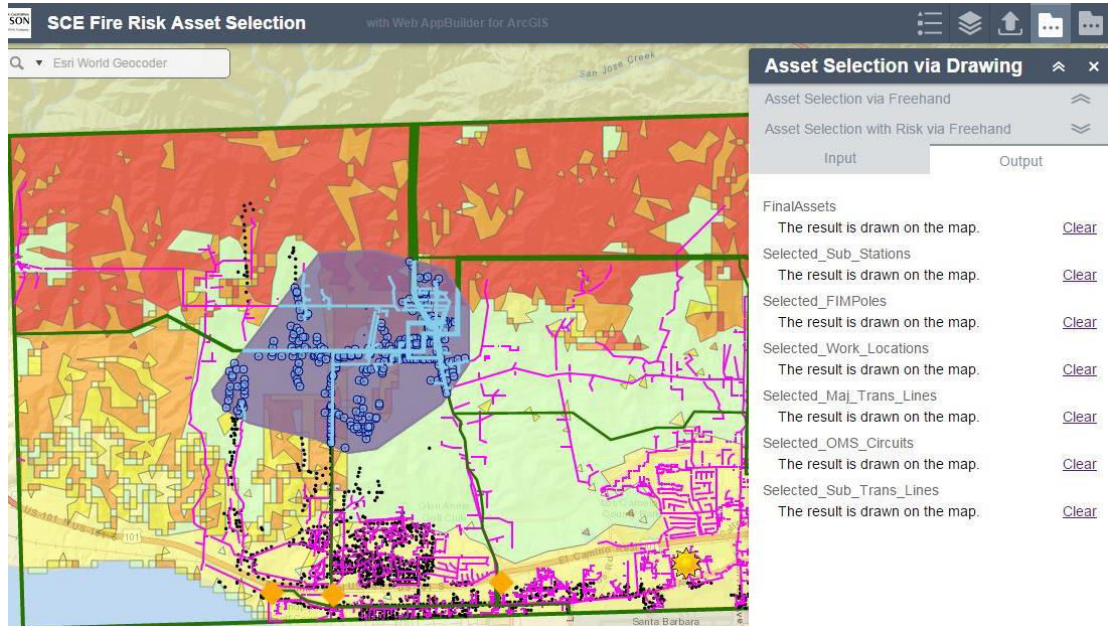
Once there, the attribute table can be opened for each layer through its options menu (Figure 6.6). In addition, all the tables can be exported as a .csv file.



**Figure 6.6: Operational Layer Menu**

In this example, 203 assets were selected, among which there were 0 work locations, 0 sub stations, 0 major transmission lines, 6 OMS circuits, and 193 FIM poles. In less than a minute, the user is ready to send the information on the assets within the area of interest to their supervisor. Using this geoprocessing tool, the client is able to search for the relevant assets in an efficient way that will in turn help them protect their valuable facilities. In a different scenario where the client may be interested in managing the assets in relation to fire risk or providing risk levels for an insurance company, the user can choose the *Asset Selection with Risk* geoprocessing tool that is located underneath the

*Asset Selection* tool on the *Asset Selection via Draw* tab (Figure 6.7). Similar to the previous tool, the user can draw the area of interest and execute the evaluation on that area.



**Figure 6.7: Asset Selection for Risk Evaluation**

The difference between the GP Tools is in the table output. This table contains risk levels for each asset based on the assets' locations and the fire risk polygons, as discussed in Chapter 5. Figure 6.8 shows the *Final Assets* table with an additional field to indicate the fire risk associated with the selected assets. These tables work the same as those in the *Asset Selection* GP tool; they can be opened and closed when necessary.

CIRCUIT_NO	NAME	KV	Asset_ID	Asset_Type	THREAT_LEVEL
01683-M	GOLETA-ELLWOOD-ISLA VISTA-ONSHORE	2	STL10	SubTransmissionLine	High
01576-M	GOLETA-CAPITAN	2	STL9	SubTransmissionLine	High
05175-M	GOLETA-CAPITAN-GAVIOTA	2	STL8	SubTransmissionLine	High
00330-M	GOLETA-SAN MARCOS	2	STL7	SubTransmissionLine	Extreme
00324-M	GOLETA-VEGAS	2	STL6	SubTransmissionLine	Extreme
00339-M	GOLETA-SANTA BARBARA	1	STL5	SubTransmissionLine	Extreme
01800-M	GOLETA-COLEGIO-ISLA VISTA-	1	STL4	SubTransmissionLine	Little or No Threat

**Figure 6.8: Asset Selection with Risk Output Table**

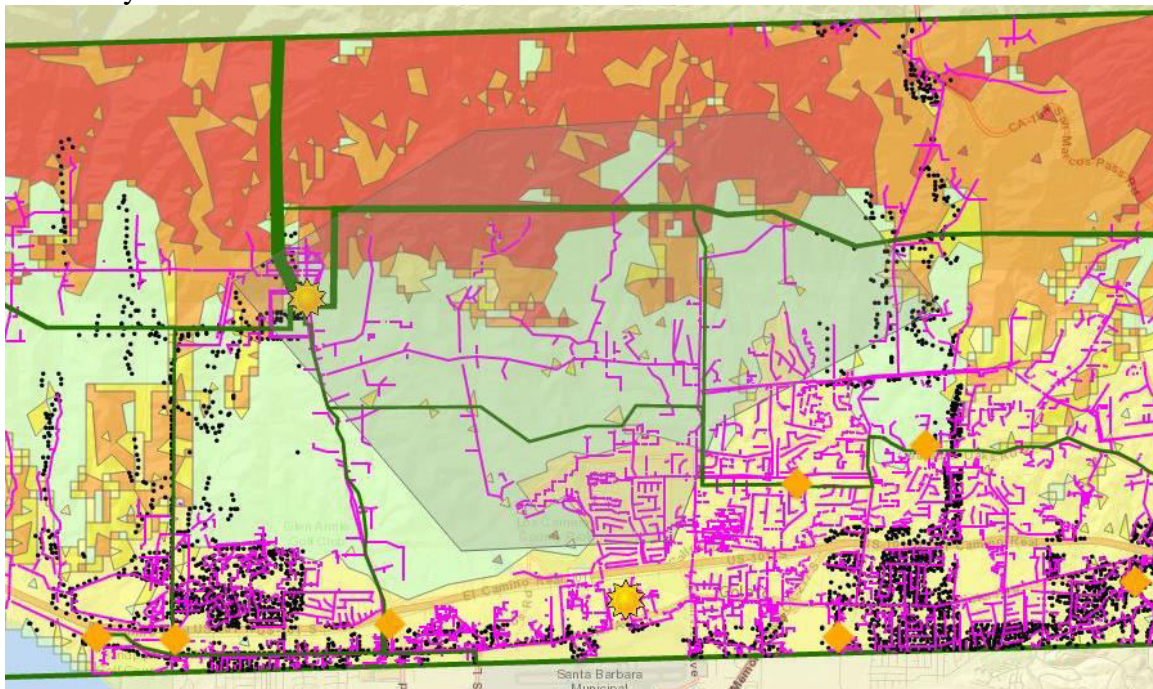
## 6.2 Asset Selection with Shapefile Uploaded

In some situations, the area of interest may be predefined. For example, an insurance company might be interested in evaluating the assets in the City of Goleta where the border is very specific. To query the assets within the city limit, the user would need to have a shapefile for the boundary of the city and upload the shapefile to the web application. Figure 6.9 shows the *Add Shape* tool interface.



**Figure 6.9: Add Shapefile to Application**

The user will click the *Choose File* button, and browse to the shapefile for the city border. Once this shapefile is added to the map (Figure 6.10), it becomes a graphical operational layer and used as the area of interest.



**Figure 6.10: Shapefile Upload**

With the city limit uploaded to the map, the user can search for all the assets in the city limit and output the associated risk levels. The *Asset Selection via Shapefile* tab is located to the far right on the tab bar. The user can choose either the *Asset Selection* or the *Asset Selection with Risk* geoprocessing tool. In this case, the *Asset Selection with Risk* would be appropriate to use. After two minutes, 1571 assets are extracted and placed in the *Final Assets* table, as well as divided into the other six tables. The user can choose to send out the *Final Assets* table or specific types of assets to the insurance company.

It is worth pointing out that in order to completely clear shapefiles from the web application, the web page must be refreshed. Hand-drawn polygons can be cleared and shapefiles can be turned off, but complete removal of the shapefiles requires a refresh.

### **6.3 Summary**

This chapter discussed the use cases of the web application and its geoprocessing tools. These use cases demonstrated each geoprocessing tool and the reasons for using them. The number of options available to the user allows the tool to serve the needs of the user without overwhelming them or the intended recipient of the information.



## Chapter 7 – Conclusions and Future Work

This project aimed to modernize Southern California Edison's methods of interacting with and evaluating its various assets. This was accomplished by writing Python scripts that searched through areas the user deemed important, and giving them multiple ways of viewing that information.

This project fulfilled the functional and non-functional requirements set forth at the beginning of the process. The scripts were designed in Python and uploaded to a web server to facilitate the creation of a web application. This web application allowed easy access to the data and GP tools for manipulating them on the internet. Each GP tool allows the user to interact with the data in specific ways, limiting information clutter. The asset and fire risk data were organized into a geodatabase for ease of access and readability. By creating a web application, HTML 5 and JavaScript were utilized to create an open source medium for interacting with data that is less restrictive than a licensed software package.

Initial versions of this tool incorporated the ArcGIS API for JavaScript and Model Builder for ArcGIS Desktop. However, these options became unsuitable for project completion within the scope. Model Builder for ArcGIS Desktop is a useful local way to string together and automate tool use. Unfortunately, Model Builder often assumes the presence of feature layers when executing. One notable example is the *Make Feature Layer/Copy Features* combination. *Copy Features* creates a duplicate of the input for the tool. This is most important for selections, allowing subset feature class creation. This works normally when used on Desktop through the Model Builder application. If this model is published to a service on the web, however, the geoprocessing tool will fail to run because once converted from a model to a GP tool, the *Copy Features* tool needs a feature layer of the selections to be created, and to make the copy of that layer. The GP tool cannot normally export selections that are still sitting on the base data. In Desktop, a feature layer is automatically created behind the scenes to accomplish the copy feature tool, a process not done on the web. This is just one example of Model Builder taking shortcuts, and this risk compromises its usefulness in transitioning to web use.

The ArcGIS API for JavaScript has a similar problem. While it allows for customization that cannot be obtained elsewhere, that customization requires specific programming to achieve. In order to execute the asset selection script discussed in Chapter 5, the script code has to be almost completely rewritten, but with a different syntax to match JavaScript instead of Python. This drastically increases the time needed to create the final product, forcing everything to be developed twice. Because of this, ArcGIS Online WebApp Builder was a far simpler and straightforward approach to meeting the requirements of the project.

Despite these inefficiencies, Model Builder and the JavaScript API for ArcGIS still provide interesting possibilities for future work. Creating models that account for every aspect of a GP tool allows for an easier workflow for the casual user than interpreting Python code. The JavaScript API also provides the largest number of customization options, and with the base functionalities created through this project, a JavaScript adaptation of these scripts could create a user interface that is more intuitive and contains greater options for manipulating the data.

Another future adaptation would be to attain the most current data on fire occurrences, vegetation types, aspect and slope information, and to create a fire analysis tool that calculates the fire risk instead of obtaining it from another source. This would increase future reusability and adaptability, especially if any of the data used in the fire analysis could be gathered on a regular basis, as this would keep risk more current and increase reliability of the risk values.



## Works Cited

- Chow, T. E. (2008), The Potential of Maps APIs for Internet GIS Applications. *Transactions in GIS*, 12: 179–191. doi: 10.1111/j.1467-9671.2008.01094.x
- Finney, Mark A. (2006). An overview of FlamMap fire modeling capabilities. Pages 213–220.
- Flanagan, D. (2002). Introduction to JavaScript. In *JavaScript: The definitive guide* (4th ed., p. 2). Sebastopol, CA: O'Reilly.
- Goldberg, G. E., Neuenschwander, L. F., & Ryan, K. C. (2001). Introduction: Integrating spatial technologies and ecological principles for a new age in fire management. *International Journal of Wildland Fire*, 10(4), 263–265.
- Gomez C., Mangeas M., Curt T., Ibanez T., Munzinger J., Dumas P., Jérémy A., Despinoy M., & Hély C., *Ecology and Evolution* 2015; 5(2): 377–390.
- Haklay, M., Singleton, A. and Parker, C. (2008), Web Mapping 2.0: The Neogeography of the GeoWeb. *Geography Compass*, 2: 2011–2039. doi: 10.1111/j.1749-8198.2008.00167.
- Keane, R. E., Burgan, R., & van Wagendonk, J. (2001). Mapping wildland fuels for fire management across multiple scales: integrating remote sensing, GIS, and biophysical modeling. *International Journal of Wildland Fire*, 10(4), 301–319.
- Robson, E., & Freeman, E. (2012). Getting to Know HTML. In *Head first HTML and CSS* (Second ed., pp. 4–6). Beijing: O'Reilly.
- Rothermel R. C. (1972) A mathematical model for predicting fire spread in wildland fuels. USDA Forest Service Research Paper INT-115.
- Sanner, M. F. (1999). Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1), 57–61.
- Scott, J. H., & Burgan, R. E. (2005). Standard fire behavior fuel models: a comprehensive set for use with Rothermel's surface fire spread model. *The Bark Beetles, Fuels, and Fire Bibliography*, 66.
- Stephens, S. L. (1998). Evaluation of the effects of silvicultural and fuels treatments on potential fire behaviour in Sierra Nevada mixed-conifer forests. *Forest Ecology and Management*, 105(1), 21–35.
- VanRossum, G., & Drake, F. L. (2010). *The Python Language Reference*. Python Software Foundation.





## Appendix A. Feature Set Schema

### FIM Poles

Attribute	Data Type
OBJECTID	Object ID
Shape	Geometry
STRUCTURE_NUMBER	String
FIM_DIST_NUM	Long
FIM_LAYER	String
COUNTY	String
AgencyName	String
Asset_ID	String
Asset_Type	String

### Substations

Attribute	Data Type
OBJECTID	Object ID
Shape	Geometry
NAME	String
SUB_TYPE	String
PRI_KV	Double
SEC_KV	Double
DIST_NAME	String
DIST_NO	String
Asset_ID	String
Asset_Type	String

### Work Locations

Attribute	Data Type
OBJECTID	Object ID
Shape	Geometry
NAME	String
WL_NO	String
TYPE	String
DIST_NAME	String
DIST_NUM	Short
Asset_ID	String
Asset_Type	String

### Sub Transmission Lines

Attribute	Data Type
OBJECTID	Object ID
Shape	Geometry
CIRCUIT_NO	String
NAME	String
KV	Double
SHAPE_Length	Double
Asset_ID	String
Asset_Type	String

### Major Transmission Lines

Attribute	Data Type
OBJECTID	Object ID
Shape	Geometry
CIRCUIT_NO	String
NAME	String
KV	Double
SHAPE_Length	Double
Asset_ID	String
Asset_Type	String

### OMS Circuits

Attribute	Data Type
OBJECTID	Object ID
Shape	Geometry
CIRCT_NAM	String
CIRCT_VOLT	Float
SHAPE_Length	Double
Asset_ID	String
Asset_Type	String

## Appendix B. Asset Selection Script

```
#-----
#-----
# Developer: Andrew Sanchez
# AssetSelectionFinal.py
# Created on: 2015-06-29
# Description:
# Use this tool to select assets and create attribute tables of the selected
#-----
#-----
# Import arcpy module
#-----
import arcpy

#-----
#-----
# Establish Environment Qualities
#-----
arcpy.env.workspace = "C:\\MIP\\FireToolData_Andrew\\FireToolData.gdb"
arcpy.env.overwriteOutput= True

#-----
#-----
# Define Variables
#-----

#-----
#Original Asset Data from Southern California Edison (SCE) and their Feature
#Layer Version
#-----
WorkLocations = "OriginalData\\WorkLocations"
WorkLocations_lyr = "in_memory\\WorkLocations_Lyr"
SubTransmissionLines = "OriginalData\\SubTransmissionLines"
SubTransmissionLines_lyr = "in_memory\\SubTransmissionLines_Lyr"
Substations = "OriginalData\\Substations"
Substations_lyr = "in_memory\\Substations_Lyr"
OMSCircuits = "OriginalData\\OMSCircuits"
OMSCircuits_lyr = "in_memory\\OMSCircuits_Lyr"
MajorTransmissionLines = "OriginalData\\MajorTransmissionLines"
MajorTransmissionLines_lyr = "in_memory\\MajorTransmissionLines_Lyr"
FIMPoles = "OriginalData\\FIMPoles"
FIMPoles_lyr = "in_memory\\FIMPoles_Lyr"
```

```

#-----
#Line Feature Classes Post Analysis in their Point Representation Form
#-----
SubTransmissionLines_Pts = "in_memory\\SubTransmissionLines_Pts"
OMSCircuits_Pts = "in_memory\\OMSCircuits_Pts"
MajorTransmissionLines_Pts = "in_memory\\MajorTransmissionLines_Pts"

#-----
#Feature class containing the Point Representations of every feature class
#Used to create the empty Final Assets Feature Class
#-----
allFeatureFC = "PointRepresentationOfAssets\\ALL_Features"

#-----
#Variables defined by the user for Input (Area of Interest) and Output Location
#-----
FeatureSetTemplate_ForTool = arcpy.GetParameterAsText(0)
if not FeatureSetTemplate_ForTool:
    FeatureSetTemplate_ForTool = "Templates\\TestPolygon"

FinalAssets = arcpy.GetParameterAsText(1)
if not FinalAssets:
    FinalAssets = "SelectedData\\FinalAssets"

#-----
#Final Output Feature Classes that hold the selected and analyzed Asset Data
#Seperated by Asset Type
#-----
Selected_Work_Locations = arcpy.GetParameterAsText(2)
if not Selected_Work_Locations:
    Selected_Work_Locations = "SelectedData\\Selected_Work_Locations"

Selected_FIMPoles = arcpy.GetParameterAsText(4)
if not Selected_FIMPoles:
    Selected_FIMPoles = "SelectedData\\Selected_FIMPoles"

Selected_Sub_Stations = arcpy.GetParameterAsText(3)
if not Selected_Sub_Stations:
    Selected_Sub_Stations = "SelectedData\\Selected_Sub_Stations"

Selected_Maj_Trans_Lines = arcpy.GetParameterAsText(7)
if not Selected_Maj_Trans_Lines:
    Selected_Maj_Trans_Lines = "SelectedData\\Selected_Maj_Trans_Lines"

Selected_OMS_Circuits = arcpy.GetParameterAsText(5)
if not Selected_OMS_Circuits:
    Selected_OMS_Circuits = "SelectedData\\Selected_OMS_Circuits"

Selected_Sub_Trans_Lines = arcpy.GetParameterAsText(6)
if not Selected_Sub_Trans_Lines:
    Selected_Sub_Trans_Lines = "SelectedData\\Selected_Sub_Trans_Lines"

#-----
#-----
# Create Empty Feature classes to hold selections with the correct Schema
#-----
#-----

#These are the same Feature classes that had their pathways defined above

```

```

arcpy.CopyFeatures_management(allFeatureFC, FinalAssets, "", "0", "0", "0")
arcpy.DeleteFeatures_management(FinalAssets)

arcpy.CopyFeatures_management(WorkLocations, Selected_Work_Locations, "", "0", "0",
"0")
arcpy.DeleteFeatures_management(Selected_Work_Locations)

arcpy.CopyFeatures_management(Substations, Selected_Sub_Stations, "", "0", "0", "0")
arcpy.DeleteFeatures_management(Selected_Sub_Stations)

arcpy.CopyFeatures_management(FIMPoles, Selected_FIMPoles, "", "0", "0", "0")
arcpy.DeleteFeatures_management(Selected_FIMPoles)

arcpy.CopyFeatures_management(MajorTransmissionLines, Selected_Maj_Trans_Lines, "",
"0", "0", "0")
arcpy.DeleteFeatures_management(Selected_Maj_Trans_Lines)

arcpy.CopyFeatures_management(SubTransmissionLines, Selected_Sub_Trans_Lines, "", "0",
"0", "0")
arcpy.DeleteFeatures_management(Selected_Sub_Trans_Lines)

arcpy.CopyFeatures_management(OMSCircuits, Selected_OMS_Circuits, "", "0", "0", "0")
arcpy.DeleteFeatures_management(Selected_OMS_Circuits)
#-----
#-----
# Use Select Layer by Location tool on variables
#-----
#-----

#Each of the six feature classes have assets selected by the area of interest
#defined by the user

# Process: Select Work Locations By Location
arcpy.MakeFeatureLayer_management(WorkLocations, WorkLocations_lyr)
arcpy.SelectLayerByLocation_management(WorkLocations_lyr, "WITHIN",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Once the assets are selected, they are made permanent in the feature classes
#created at the beginning
arcpy.CopyFeatures_management(WorkLocations_lyr, Selected_Work_Locations, "", "0", "0",
"0")

#-----
#-----
# Process: Select FIM Poles By Location
arcpy.MakeFeatureLayer_management(FIMPoles, FIMPoles_lyr)
arcpy.SelectLayerByLocation_management(FIMPoles_lyr, "WITHIN",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Once the assets are selected, they are made permanent in the feature classes
#created at the beginning
arcpy.CopyFeatures_management(FIMPoles_lyr, Selected_FIMPoles, "", "0", "0", "0")

#-----
#-----
# Process: Select Substations By Location
arcpy.MakeFeatureLayer_management(Substations, Substations_lyr)
arcpy.SelectLayerByLocation_management(Substations_lyr, "WITHIN",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Once the assets are selected, they are made permanent in the feature classes

```

```

#created at the beginning
arcpy.CopyFeatures_management(Substations_lyr, Selected_Sub_Stations, "", "0", "0",
"0")

#-----
#-----
# Process: Select Sub Trans Lines By Location
arcpy.MakeFeatureLayer_management(SubTransmissionLines, SubTransmissionLines_lyr)
arcpy.SelectLayerByLocation_management(SubTransmissionLines_lyr, "INTERSECT",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Once the assets are selected, they are made permanent in the feature classes
#created at the beginning
arcpy.CopyFeatures_management(SubTransmissionLines_lyr, Selected_Sub_Trans_Lines, "",
"0", "0", "0")

#For the line features, in order to be appended to the Final Assets Feature
#Class, they must first be converted into point features so that all six
#feature classes have identical geometry
arcpy.FeatureToPoint_management(SubTransmissionLines_lyr,SubTransmissionLines_Pts,
"CENTROID")

#-----
#-----
# Process: Select OMS Circuits By Location
arcpy.MakeFeatureLayer_management(OMSCircuits, OMSCircuits_lyr)
arcpy.SelectLayerByLocation_management(OMSCircuits_lyr, "INTERSECT",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Once the assets are selected, they are made permanent in the feature classes
#created at the beginning
arcpy.CopyFeatures_management(OMSCircuits_lyr, Selected_OMS_Circuits, "", "0", "0",
"0")

#For the line features, in order to be appended to the Final Assets Feature
#Class, they must first be converted into point features so that all six
#feature classes have identical geometry
arcpy.FeatureToPoint_management(OMSCircuits_lyr,OMSCircuits_Pts, "CENTROID")

#-----
#-----
# Process: Select Maj Trans Lines By Location
arcpy.MakeFeatureLayer_management(MajorTransmissionLines, MajorTransmissionLines_lyr)
arcpy.SelectLayerByLocation_management(MajorTransmissionLines_lyr, "INTERSECT",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Once the assets are selected, they are made permanent in the feature classes
#created at the beginning
arcpy.CopyFeatures_management(MajorTransmissionLines_lyr, Selected_Maj_Trans_Lines, "",
"0", "0", "0")

#For the line features, in order to be appended to the Final Assets Feature
#Class, they must first be converted into point features so that all six
#feature classes have identical geometry
arcpy.FeatureToPoint_management(MajorTransmissionLines_lyr,MajorTransmissionLines_Pts,
"CENTROID")

#-----
#-----
-----
# Append all Values Together

```

```
#-----
#-----

#All of the point assets and the point representations of the line assets are
#appended into the Final Assets Feature Class for convenient storage
arcpy.Append_management([Selected_FIMPoles, MajorTransmissionLines_Pts,
OMSCircuits_Pts, Selected_Sub_Stations, SubTransmissionLines_Pts,
Selected_Work_Locations], FinalAssets, "NO_TEST", "", "")
```

## Appendix C. Asset Selection with Risk Script

```
#-----
#-----
# Developer: Andrew Sanchez
# AssetSelectionFinal.py
# Created on: 2015-06-29
# Description:
# Use this tool to select assets and create attribute tables of the selected
#-----
#-----
# Import arcpy module
#-----
#-----
import arcpy

#-----
#-----
# Establish Environment Qualities
#-----
#-----
arcpy.env.workspace = "C:\\MIP\\FireToolData_Andrew\\FireToolData.gdb"
arcpy.env.overwriteOutput= True

#-----
#-----
# Define Variables
#-----
#-----

#-----
#Polygons That hold the Fire Risk Values
#-----
Fire_Threat_Polygons = "OriginalData\\Fire_Threat_Polygons"

#-----
#Original Asset Data from Southern California Edison (SCE) and their Feature
#Layer Version
#-----
WorkLocations = "OriginalData\\WorkLocations"
WorkLocations_lyr = "in_memory\\WorkLocations_Lyr"
SubTransmissionLines = "OriginalData\\SubTransmissionLines"
SubTransmissionLines_lyr = "in_memory\\SubTransmissionLines_Lyr"
Substations = "OriginalData\\Substations"
Substations_lyr = "in_memory\\Substations_Lyr"
OMSCircuits = "OriginalData\\OMSCircuits"
OMSCircuits_lyr = "in_memory\\OMSCircuits_Lyr"
MajorTransmissionLines = "OriginalData\\MajorTransmissionLines"
MajorTransmissionLines_lyr = "in_memory\\MajorTransmissionLines_Lyr"
```



```

FIMPoles = "OriginalData\\FIMPoles"
FIMPoles_lyr = "in_memory\\FIMPoles_lyr"

#-----
#Line Feature Classes Post Analysis in their Point Representation Form
#-----
SubTransmissionLines_Pts = "in_memory\\SubTransmissionLines_Pts"
OMSCircuits_Pts = "in_memory\\OMSCircuits_Pts"
MajorTransmissionLines_Pts = "in_memory\\MajorTransmissionLines_Pts"

#-----
#Point Feature Classes containing Fire Risk Attributes alongside the default
#descriptive attributes of the features
#-----
SpatialFIM = "in_memory\\SpatialFIM"
SpatialSubSt = "in_memory\\SpatialSubSt"
SpatialWrkLoc = "in_memory\\SpatialWrkLoc"

#-----
#Feature class containing the Point Representations of every feature class
#Used to create the empty Final Assets Feature Class
#-----
allFeatureFC = "PointRepresentationOfAssets\\ALL_Features"

#-----
#The Portions of the Line Feature Classes Which Fall Within the Area of Interest
#-----
SubTranLnClip = "in_memory\\SubTranLnClip"
MajTranLnClip = "in_memory\\MajTranLnClip"
OMSClip = "in_memory\\OMSClip"

#-----
#Variables involved in the Summary Statistics performed on the Sub Transmission
#Lines to determine what risk level should be assigned
#-----
SubTransSum = "in_memory\\SubTransSum"
SubTranMax = "in_memory\\SubTranMax"
SubTranLnClip_lyr = "in_memory\\SubTranLnClip_lyr"
SubTranLnIntersect = "in_memory\\SubTranLnIntersect"

#-----
#Variables involved in the Summary Statistics performed on the Major
#Transmission Lines to determine what risk level should be assigned
#-----
MajTranSum = "in_memory\\SubTransSum"
MajTranMax = "in_memory\\SubTranMax"
MajTranClip_lyr = "in_memory\\SubTranLnClip_lyr"
MajTranIntersect = "in_memory\\SubTranLnIntersect"

#-----
#Variables involved in the Summary Statistics performed on the OMS Circuits
#to determine what risk level should be assigned
#-----
OMSSum = "in_memory\\SubTransSum"
OMSMMax = "in_memory\\SubTranMax"
OMSClip_lyr = "in_memory\\SubTranLnClip_lyr"
OMSIntersect = "in_memory\\SubTranLnIntersect"

#-----
#Variables defined by the user for Input (Area of Interest) and Output Location

```

```

#-----
FeatureSetTemplate_ForTool = arcpy.GetParameterAsText(0)
if not FeatureSetTemplate_ForTool:
    FeatureSetTemplate_ForTool = "Templates\\TestPolygon"

FinalAssets = arcpy.GetParameterAsText(1)
if not FinalAssets:
    FinalAssets = "SelectedData\\FinalAssets"

#-----
#Final Output Feature Classes that hold the selected and analyzed Asset Data
#Seperated by Asset Type
#-----
Selected_Work_Locations = arcpy.GetParameterAsText(2)
if not Selected_Work_Locations:
    Selected_Work_Locations = "SelectedData\\Selected_Work_Locations"

Selected_FIMPoles = arcpy.GetParameterAsText(4)
if not Selected_FIMPoles:
    Selected_FIMPoles = "SelectedData\\Selected_FIMPoles"

Selected_Sub_Stations = arcpy.GetParameterAsText(3)
if not Selected_Sub_Stations:
    Selected_Sub_Stations = "SelectedData\\Selected_Sub_Stations"

Selected_Maj_Trans_Lines = arcpy.GetParameterAsText(7)
if not Selected_Maj_Trans_Lines:
    Selected_Maj_Trans_Lines = "SelectedData\\Selected_Maj_Trans_Lines"

Selected_OMS_Circuits = arcpy.GetParameterAsText(5)
if not Selected_OMS_Circuits:
    Selected_OMS_Circuits = "SelectedData\\Selected_OMS_Circuits"

Selected_Sub_Trans_Lines = arcpy.GetParameterAsText(6)
if not Selected_Sub_Trans_Lines:
    Selected_Sub_Trans_Lines = "SelectedData\\Selected_Sub_Trans_Lines"

#-----
#-----
# Create Empty Feature classes to hold selections with the correct Schema
#-----
#-----

#These are the same Feature classes that had their pathways defined above

arcpy.CopyFeatures_management(allFeatureFC, FinalAssets, "", "0", "0", "0")
arcpy.DeleteFeatures_management(FinalAssets)

arcpy.CopyFeatures_management(WorkLocations, Selected_Work_Locations, "", "0", "0",
"0")
arcpy.DeleteFeatures_management(Selected_Work_Locations)

arcpy.CopyFeatures_management(Substations, Selected_Sub_Stations, "", "0", "0", "0")
arcpy.DeleteFeatures_management(Selected_Sub_Stations)

arcpy.CopyFeatures_management(FIMPoles, Selected_FIMPoles, "", "0", "0", "0")
arcpy.DeleteFeatures_management(Selected_FIMPoles)

arcpy.CopyFeatures_management(MajorTransmissionLines, Selected_Maj_Trans_Lines, "",
"0", "0", "0")
arcpy.DeleteFeatures_management(Selected_Maj_Trans_Lines)

```

```

arcpy.CopyFeatures_management(SubTransmissionLines, Selected_Sub_Trans_Lines, "", "0",
"0", "0")
arcpy.DeleteFeatures_management(Selected_Sub_Trans_Lines)

arcpy.CopyFeatures_management(OMSCircuits, Selected_OMS_Circuits, "", "0", "0", "0")
arcpy.DeleteFeatures_management(Selected_OMS_Circuits)

#-----
#-----
# Add Necessary Field to House Fire Risk Attributes
#-----
#-----

#The Final Assets Feature Class does not contain a Threat Level Field by default
#but needs one to house the incoming data
arcpy.AddField_management(FinalAssets, "THREAT_LEVEL", "TEXT")

#-----
#-----
# Use Select Layer by Location tool on variables
#-----
#-----

#Each of the six feature classes have assets selected by the area of interest
#defined by the user

# Process: Select Work Locations By Location
arcpy.MakeFeatureLayer_management(WorkLocations, WorkLocations_lyr)
arcpy.SelectLayerByLocation_management(WorkLocations_lyr, "WITHIN",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Spatial Join is used to unite the point assets with their corresponding risk
#levels based on their location
arcpy.SpatialJoin_analysis(WorkLocations_lyr, Fire_Threat_Polygons, SpatialWrkLoc,
"JOIN_ONE_TO_ONE", "KEEP_ALL", "", "WITHIN", "", "")

#Once the assets and their risk levels are joined, they are made permanent
#in the feature classes created at the beginning
arcpy.CopyFeatures_management(SpatialWrkLoc, Selected_Work_Locations, "", "0", "0",
"0")

#-----
#-----
# Process: Select FIM Poles By Location
arcpy.MakeFeatureLayer_management(FIMPoles, FIMPoles_lyr)
arcpy.SelectLayerByLocation_management(FIMPoles_lyr, "WITHIN",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Spatial Join is used to unite the point assets with their corresponding risk
#levels based on their location
arcpy.SpatialJoin_analysis(FIMPoles_lyr, Fire_Threat_Polygons, SpatialFIM,
"JOIN_ONE_TO_ONE", "KEEP_ALL", "", "WITHIN", "", "")

#Once the assets and their risk levels are joined, they are made permanent
#in the feature classes created at the beginning
arcpy.CopyFeatures_management(SpatialFIM, Selected_FIMPoles, "", "0", "0", "0")

#-----
#-----
# Process: Select Substations By Location

```

```

arcpy.MakeFeatureLayer_management(Substations, Substations_lyr)
arcpy.SelectLayerByLocation_management(Substations_lyr, "WITHIN",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Spatial Join is used to unite the point assets with their corresponding risk
#levels based on their location
arcpy.SpatialJoin_analysis(Substations_lyr, Fire_Threat_Polygons, SpatialSubSt,
"JOIN_ONE_TO_ONE", "KEEP_ALL", "", "WITHIN", "", "")

#Once the assets and their risk levels are joined, they are made permanent
#in the feature classes created at the beginning
arcpy.CopyFeatures_management(SpatialSubSt, Selected_Sub_Stations, "", "0", "0", "0")

#-----
#-----
# Process: Select Sub Trans Lines By Location
arcpy.MakeFeatureLayer_management(SubTransmissionLines, SubTransmissionLines_lyr)
arcpy.SelectLayerByLocation_management(SubTransmissionLines_lyr, "INTERSECT",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Lines have to be treated differently from points. Instead of Spatial Join
#the line features are clipped to the area of interest and intersected with the
#Fire Risk Polygons to receive their fire risk levels
arcpy.Clip_analysis(SubTransmissionLines_lyr, FeatureSetTemplate_ForTool,
SubTranLnClip)
arcpy.Intersect_analysis([SubTranLnClip, Fire_Threat_Polygons], SubTranLnIntersect,
"ALL", "", "INPUT")
arcpy.AddField_management(SubTranLnIntersect, "Shape_Length", "DOUBLE")
arcpy.CalculateField_management(SubTranLnIntersect, "Shape_Length", "!SHAPE.Length!",
"PYTHON_9.3")

#Once the assets have been intersected with their fire risk levels, a summary
#statistics is applied to the clipped lines to determine which fire risk level
#most influences the asset
#Then a Max statistic is applied in order to export the records that hold the
#most influence over the assets
arcpy.Statistics_analysis(SubTranLnIntersect, SubTransSum, [{"Shape_Length", "SUM"}],
["Asset_ID", "THREAT_LEVEL"])
arcpy.Statistics_analysis(SubTransSum, SubTranMax, [{"SUM_Shape_Length", "MAX"}],
["Asset_ID", "THREAT_LEVEL"])

#After creating a table that contains the most relevant risk levels for each
#line feature, those values are joined back to the line feature classes clipped
#to the area of interest
#These assets are then copied over to their permanent location in the feature
#classes designed at the start
arcpy.MakeFeatureLayer_management(SubTranLnClip, SubTranLnClip_lyr)
arcpy.JoinField_management(SubTranLnClip_lyr, "Asset_ID", SubTranMax, "Asset_ID",
"THREAT_LEVEL")
arcpy.CopyFeatures_management(SubTranLnClip_lyr, Selected_Sub_Trans_Lines, "", "0",
"0", "0")

#The attributes within the final feature classes is converted into points
#in order to allow them to be appended into the Final Assets Feature Class
arcpy.FeatureToPoint_management(Selected_Sub_Trans_Lines, SubTransmissionLines_Pts,
"CENTROID")

#-----
#-----
# Process: Select OMS Circuits By Location
arcpy.MakeFeatureLayer_management(OMSCircuits, OMSCircuits_lyr)

```

```

arcpy.SelectLayerByLocation_management(OMSCircuits_lyr, "INTERSECT",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Lines have to be treated differently from points. Instead of Spatial Join
#the line features are clipped to the area of interest and intersected with the
#Fire Risk Polygons to receive their fire risk levels
arcpy.Clip_analysis(OMSCircuits_lyr, FeatureSetTemplate_ForTool, OMSClip)
arcpy.Intersect_analysis([OMSClip, Fire_Threat_Polygons], OMSIntersect, "ALL", "",
"INPUT")
arcpy.AddField_management(OMSIntersect, "Shape_Length", "DOUBLE")
arcpy.CalculateField_management(OMSIntersect, "Shape_Length", "!SHAPE.Length!",
"PYTHON_9.3")

#Once the assets have been intersected with their fire risk levels, a summary
#statistics is applied to the clipped lines to determine which fire risk level
#most influences the asset
#Then a Max statistic is applied in order to export the records that hold the
#most influence over the assets
arcpy.Statistics_analysis(OMSIntersect, OMSSum, [{"Shape_Length", "SUM"}], ["Asset_ID",
"THREAT_LEVEL"])
arcpy.Statistics_analysis(OMSSum, OMSMax, [{"SUM_Shape_Length", "MAX"}], ["Asset_ID",
"THREAT_LEVEL"])

#After creating a table that contains the most relevant risk levels for each
#line feature, those values are joined back to the line feature classes clipped
#to the area of interest
#These assets are then copied over to their permanent location in the feature
#classes designed at the start
arcpy.MakeFeatureLayer_management(OMSClip, OMSClip_lyr)
arcpy.JoinField_management(OMSClip_lyr, "Asset_ID", OMSMax, "Asset_ID", "THREAT_LEVEL")
arcpy.CopyFeatures_management(OMSClip_lyr, Selected_OMS_Circuits, "", "0", "0", "0")

#The attributes within the final feature classes is converted into points
#in order to allow them to be appended into the Final Assets Feature Class
arcpy.FeatureToPoint_management(Selected_OMS_Circuits, OMSCircuits_Pts, "CENTROID")

#-----
#-----
# Process: Select Maj Trans Lines By Location
arcpy.MakeFeatureLayer_management(MajorTransmissionLines, MajorTransmissionLines_lyr)
arcpy.SelectLayerByLocation_management(MajorTransmissionLines_lyr, "INTERSECT",
FeatureSetTemplate_ForTool, "", "NEW_SELECTION")

#Lines have to be treated differently from points. Instead of Spatial Join
#the line features are clipped to the area of interest and intersected with the
#Fire Risk Polygons to receive their fire risk levels
arcpy.Clip_analysis(MajorTransmissionLines_lyr, FeatureSetTemplate_ForTool,
MajTranLnClip)
arcpy.Intersect_analysis([MajTranLnClip, Fire_Threat_Polygons], MajTranIntersect,
"ALL", "", "INPUT")
arcpy.AddField_management(MajTranIntersect, "Shape_Length", "DOUBLE")
arcpy.CalculateField_management(MajTranIntersect, "Shape_Length", "!SHAPE.Length!",
"PYTHON_9.3")

#Once the assets have been intersected with their fire risk levels, a summary
#statistics is applied to the clipped lines to determine which fire risk level
#most influences the asset
#Then a Max statistic is applied in order to export the records that hold the
#most influence over the assets
arcpy.Statistics_analysis(MajTranIntersect, MajTranSum, [{"Shape_Length", "SUM"}],
["Asset_ID", "THREAT_LEVEL"])

```

```

arcpy.Statistics_analysis(MajTranSum, MajTranMax, [{"SUM_Shape_Length", "MAX"}],
["Asset_ID", "THREAT_LEVEL"])

#After creating a table that contains the most relevant risk levels for each
#line feature, those values are joined back to the line feature classes clipped
#to the area of interest
#These assets are then copied over to their permanent location in the feature
#classes designed at the start
arcpy.MakeFeatureLayer_management(MajTranLnClip, MajTranClip_lyr)
arcpy.JoinField_management(MajTranClip_lyr, "Asset_ID", MajTranMax, "Asset_ID",
"THREAT_LEVEL")
arcpy.CopyFeatures_management(MajTranClip_lyr, Selected_OMS_Circuits, "", "0", "0",
"0")

#The attributes within the final feature classes is converted into points
#in order to allow them to be appended into the Final Assets Feature Class
arcpy.FeatureToPoint_management(Selected_Maj_Trans_Lines, MajorTransmissionLines_Pts,
"CENTROID")

#-----
#-----
# Append all Values Together
#-----
#-----

#All of the point assets and the point representations of the line assets are
#appended into the Final Assets Feature Class for convenient storage
arcpy.Append_management([Selected_FIMPoles, MajorTransmissionLines_Pts,
OMSCircuits_Pts, Selected_Sub_Stations, SubTransmissionLines_Pts,
Selected_Work_Locations], FinalAssets, "NO_TEST", "", "")

```