 **CS 480**

Central Washington University

A Data Visualization Application

Authors: Christopher Cottle- Project Lead

Scott Thomas- Documentation Lead

Antonio Castaneda- Lead Designer

Thomas Mathan- Quality Assurance Lead

Date: December 8, 2014

Client: Dr. Boris Kovalerchuk

Advisor: Dr. Boris Kovalerchuk

# Revision History

# Iteration #1

|  |  |  |
| --- | --- | --- |
| **Revision Date** | **Changes(s) Made** | **Name** |
| 10/22/2014 | Created Document | Scott Thomas |
| 10/28/2014 | Laid out document Structure | Christopher Cottle |
| 10/29/2014 | Added Revision and Project Management | Antonio Castaneda |
| 10/30/2014 | Added more content to Project Management | Antonio Castaneda |
| 11/2/2014 | Added content to Project Overview | Christopher Cottle |
| 11/2/2014 | Created Logo, Title Page and Introduction | Scott Thomas |
| 11/8/2014 | Added Use Cases + Requirements | Team |
| 11/8/2014 | Added Feasibility Section | Christopher Cottle |
| 11/9/2014 | Cleaned up Feasibility Section | Christopher Cottle |
| 11/10/2014 | Testing | Tommy Mathan |
| 11/10/2014 | UI and conclusion | Team |
| 11/10/2014 | Iteration #1 Review and clean up | Team |
| 11/26/2014 | Started Iteration #2 | Team |
| 12/1/2014 | Added Software Design Section | Christopher Cottle |
| 12/4/2014 | Added Architectural Design & part of Design Patterns | Antonio Castaneda |
| 12/8/2014 | Added Hardware Requirements, Updated U/I, Updated Conclusion | Scott Thomas |
| 12/8/2014 | Added, site map, website design, Gantt chart | Thomas Mathan |
| 12/8/2014 | Finished up design patterns, | Chris Cottle |
| 12/8/2014 | Retrospective, conclusion updated | Team |

# 

# 

# Table of Contents

Contents

[Revision History 2](#_Toc405805800)

[Iteration #1 2](#_Toc405805801)

[Table of Contents 3](#_Toc405805802)

[Table of Figures 4](#_Toc405805803)

[Introduction 4](#_Toc405805804)

[Project Overview 6](#_Toc405805805)

[People 6](#_Toc405805806)

[Background 6](#_Toc405805807)

[Justification 7](#_Toc405805808)

[Project Management 8](#_Toc405805809)

[Process Model 8](#_Toc405805810)

[Risk Management 10](#_Toc405805811)

[Software Engineering Tools 11](#_Toc405805812)

[Requirements 12](#_Toc405805813)

[Hardware and Software Requirements 12](#_Toc405805814)

[Functional/Nonfunctional Requirements 13](#_Toc405805815)

[Use Cases 15](#_Toc405805816)

[Use case diagram for user: 17](#_Toc405805817)

[Feasibility 17](#_Toc405805818)

[Design 18](#_Toc405805819)

[Architectural Design 18](#_Toc405805820)

[User Interface Design 20](#_Toc405805821)

[Software Design 21](#_Toc405805822)

[Design Patterns 23](#_Toc405805823)

[Website Design 25](#_Toc405805824)

[File Structure 26](#_Toc405805825)

[Quality Assurance 27](#_Toc405805826)

[Document standards 27](#_Toc405805827)

[Coding standards 27](#_Toc405805828)

[Change Control Process 28](#_Toc405805829)

[Testing Process 28](#_Toc405805830)

[Conclusion 29](#_Toc405805831)

[Summary 29](#_Toc405805832)

[Retrospective 30](#_Toc405805833)

[Status Report 30](#_Toc405805834)

[Appendices 32](#_Toc405805835)

[Appendix A 32](#_Toc405805836)

[Appendix B 33](#_Toc405805837)

# Table of Figures

[Figure 1 Left to right: Thomas, Antonio, Scott, Chris 9](#_Toc405805787)

[Figure 2 Use case diagram for user 17](#_Toc405805788)

[Figure 3 High level design 19](#_Toc405805789)

[Figure 4 Parallel Coordinates 20](#_Toc405805790)

[Figure 5 Data table 20](#_Toc405805791)

[Figure 6 Menu 20](#_Toc405805792)

[Figure 7 SS diagram for web client 22](#_Toc405805793)

[Figure 8 SS diagram for Windows standalone 22](#_Toc405805794)

[Figure 9 MVD facade pattern 24](#_Toc405805795)

[Figure 10 MVD command pattern(web) 24](#_Toc405805796)

[Figure 11 MVD command pattern (standalone) 25](#_Toc405805797)

[Figure 12 Site map 26](#_Toc405805798)

[Figure 13 MVD Input/Outputs 27](#_Toc405805799)

# Introduction

*This section serves to introduce the core motivation for producing the visualization application and presents our initial vision for the final product. Also it briefly addresses the purpose of this document and who can benefit from reading it.*

In recent years the concept of “Big data” has been growing in popularity, to the point that there is now an entire industry built around its collection and analysis. With advances in data collection and collation techniques, researchers have access to complex data about previously undocumented relationships within systems. However the complexity of the data presents a new challenge; the data must be represented such a way that a human can extract patterns from it.

This project implements several methods of presenting complex data, specifically multidimensional data. These methods, developed by Dr. Boris Kovalerchuk, are characterized by graphing algorithmically paired points of multidimensional data on a single formatted two dimensional plane. The representation does not reduce the data the way many other visualization techniques do and thus provides all the data necessary in a lossless format to reconstruct the original data set. By structuring the graphical representation in non-intuitive ways, patterns that would otherwise be hidden amongst the mass of data points can emerge.

The application developed by our team will transform numerical data taken from the user into as many as six different representational visualizations. In addition to the full view of the data, users will be able to trace single data elements through the visualization and compare those elements to the average case or extremes. To account for different use cases the application will be developed across three platforms: Windows 7, Web, and Excel.

This document details our plan for implementing such an application, including but not limited to: the project’s goals, how we intend to budget our time, risks we are taking on, the application requirements, the architectural structure of the application and much more. Ultimately any developer reading this document should be able to implement our application, how it works and how to fix it should anything go wrong. In addition to developers our CS480 instructor and advisor may benefit from understanding our process.

# Project Overview

*This section details the client, stakeholders and intended users of our system. It explains background information on the general factors that affect the product and its requirements. This section also gives a complete description of the problem at hand, a justification for computerizing a solution to this problem and describes the main features of the proposed system.*

## People

In the interest of successfully developing a solution to the issue of modelling multi-dimensional data, several groups of individuals must first be categorized and identified. The three groups of individuals, of which this project is developed around are: end users, project stakeholders and the client.

The client is the individual responsible for identifying a need for and requesting this system. This project’s client is Boris Kovalerchuk, Professor and Director of the Imaging Lab at Central Washington University (CWU).

The Stakeholder group is a collection of individuals with an interest or concern in the client’s requested system. In this case, stakeholders include members of the project team, the client and the CS480 course instructor. Respectively, these individuals are; Antonio Castaneda, Scott Thomas, Chris Cottle and Thomas Mathan as the project team, Boris Kovalerchuk as the client and Dr. John Anvik as CWU’s CS480 course instructor. The stakeholders have a vested interest in the project’s completion and will be communicating weekly in order to facilitate a smooth series of development cycles.

End users are the individuals that will be using the system after its completion. Since this project has both local and web deployments scheduled, our end users group could include anybody. Realistically, our end users will probably be individuals in the sciences who wish to visualize a set of multidimensional data.

## Background

Due to its nature, multi-dimensional data has been difficult to conceptualize using traditional 2 dimensional methods. As a simple example, consider a population graph for an unspecified country over a period of 100 years. The graph itself contains explicit numerical detail as to the total population of the country. One may infer the rate of population growth from a glance but in order to clearly visualize the rate of population growth it would be best to view a separate graph detailing the derivative of the first. Furthermore, if one wished to visualize the rate of change of the second graph in order to determine if the population of this country is growing at an accelerated, decelerated or constant rate, one would traditionally view yet another graph detailing the second derivative of this country’s population over the specified time period. Herein lies the problem, in order to visualize these three dimensions of data, one is required to either consult separate graphs or combine these graphs in a messy affair. Additionally, as the numbers of data dimension grow the difficulty in visualizing and understanding what they mean grows as well. Being required to consult multiple graphs to traverse each dimension of data is undesirable because it results in confusion and is difficult to see relationships between each dimension.

Although there are already methods for viewing multidimensional data across a single 2-dimensional graph, many of them are ‘lossy’ in that one cannot reconstruct the original dataset from the multidimensional graph. In the process of combining these data, it is often required that the integrity of some of that data is destroyed in order to make it visually presentable. The goal of this project is to visualize multidimensional data in a 2-dimensional medium utilizing lossless methods developed by Dr. Kovalerchuk.

## Justification

A computerized solution to this problem is the most obvious approach. Due to the nature of the datasets being parsed, computerizing this issue will provide a fast, accurate and detailed means of viewing large and sometimes tedious amount of data. The amount of time it would take a human to create and deploy a lossless 2-dimensional graph of multidimensional data would be overwhelming, this is further compacted by the fact that many of these graphs turn out to be ‘useless’ in that it is not easy for a human to readily recognize valuable information such as trending and grouping. It is often necessary to view one’s dataset in several different formats, as the nature of some trends lend themselves to being visualized more readily in certain archetypes compared to others depending on how exactly that trend is structured.

Given the problems described and the nature of the task, our program needs to be able to accept or ‘read’ a multidimensional dataset and verify its integrity. For verified datasets, the program will provide the user with several different visualization options, defaulting to a screen where it displays an initial graph. From this initial screen, the user will have the option to switch to another visualization type or load a different dataset. While viewing any graph, the user will have the ability to interact with the graph by articulating their mouse over any of the graph’s lines. While a line is being ‘hovered’ over, it will animate in some way to let the user know that they are interacting with it. These animations may include color changes, fading *other* lines away (to gain a clearer view of the selected line set), or initiating popups that will display the selected data.

This project is scoped to include a local program for Windows and a component that will perform the same functions from a web browser. Additionally, a plugin for Excel has been discussed and is currently being investigated. If this plugin is determined to be feasible, it will allow users to highlight their multidimensional dataset within excel and click a button to create one of several graphs that are available in the standalone program.

# Project Management

*In this section we will explain how we plan to manage our project. We will explain our process model and why we will be using it, a risk management plan that will describe the risks that we will be facing and how we plan to overcome them, the roles of each team member and lastly the engineering tools that we plan on using.*

## Process Model

Our team plans on using an agile development for several reasons. The majority of the requirements are clear; however, we believe these requirements may change as we implement the project. Further reasons for choosing an agile development process include:

* The total requirements of the project have not been concluded
* Some of the requirements are still vague
* Each iteration will produce a better version of the software
* Important features can be implemented first

Additionally, our team will make good use of standard agile practices. These practices include:

* User stories and use case
* Continuous Integration
* Weekly scrum meetings
* Sprint planning and reviewing
* Estimation of sprint items through comparison of previously completed tasks
* Sprint backlog
* Sprint Retrospective

We believe these practices will help not only drive forward the development of our project but also help build quality software.

**Team Member Roles**

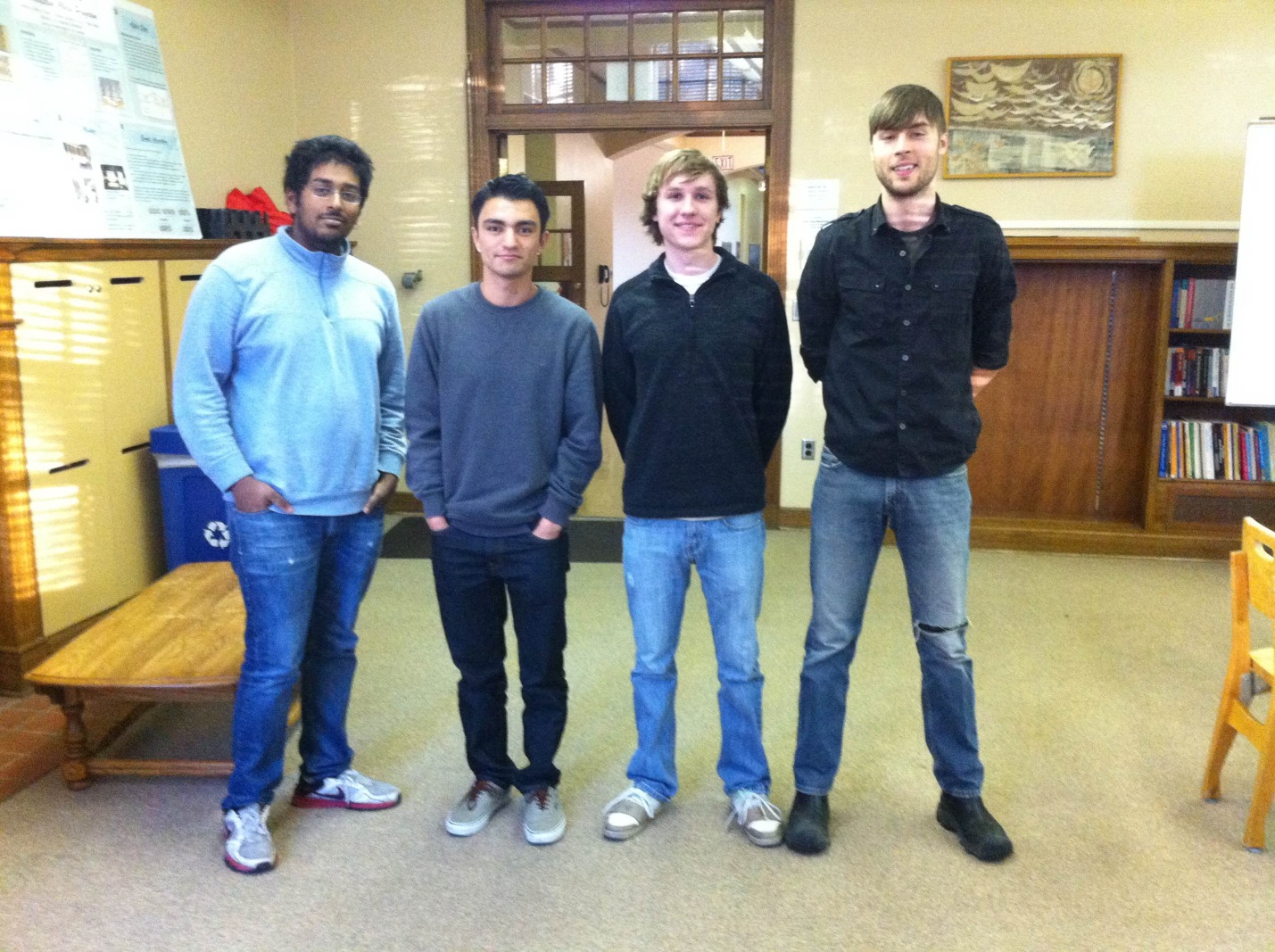
****

Figure 1 Left to right: Thomas, Antonio, Scott, Chris

Chris Cottle - Project Lead

* Set up meetings - great communication skills
* Coordinate team - excellent for helping team move forward
* Lay down the law - emphasizes tasks
* Contribute with Documentation

Scott Thomas - Documentation Lead

* Assignment submission - great ability to remember
* Reviews documentation - great writer
* Problem solver - quick at generating ideas to solve problems
* Contribute with Documentation

Antonio Castaneda - Lead Designer

* Software design - experience with software design and structure
* Familiar with OpenGL and graphics in general
* Create UML diagrams - experience with UML diagrams
* Contribute with Documentation

Thomas Mathan - Quality Assurance Lead

* Contribute with Documentation
* Writing tests and Testing - experience with testing and writing test cases
* Has great attention to detail, encourages higher quality production

## Risk Management

Though many things of this project seem feasible, there are still risks in some requirements of this project. We will ensure to evaluate these risks and do the most to overcome them.

One of the major risks is developing an add-in to Microsoft Excel. The add-in is to work within Excel and be able to display different types of graphs from selected data cells; that is being able to draw graphs through Excel using Visual Basic or a C# plugin. With preliminary research, our team decided that this is a great risk due to the difficulty of creating such an add-in. We plan on doing a great amount of research on Excel’s add-ins and how they can be implemented to overcome this obstacle. Also, if this task is not possible with one add-in, then we will create multiple, one for each different type of graph.

Another risk we face is the implementation of a web-based application. The client, Dr. Kovalerchuk, requires us to make a web-based application that is not based on Java because Java has many restrictions with access privileges. Furthermore, the web based application is not to rely on a plug-in. We plan on using a Unity engine web based application; however, it currently needs a plug-in. An update is to come soon to make the Unity engine not have a plug-in and instead work with WebGL; therefore the risk becomes that the update may never be deployed, resulting in the Unity web application needing a web plug-in. To address this risk we plan on waiting to the date of the update release which is December. If the update isn’t released, then we will implement the web based application with just WebGL.

The following table provides possible risks the team will face:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk** | **Priority** | **Likelihood** | **Impact** | **Mitigation Strategy** |
| Excel add-in | Medium | High | Medium | * Do Excel online tutorials * Read Excel API * Try to create sample add-ins |
| Web-app without plugin | High | High | Medium | * Learn WebGL * Provide easy to use plug-in * Research for alternatives |
| Member fails CS480 | High | Low | High | * Redistribute work |
| Underestimate Requirements | High | Low | Medium | * Reevaluate requirements * Redistribute work/tasks |
| Change of requirements | High | High | High | * Reevaluate requirements * Redistribute work * Set new timeline |
| Conflicting Schedules | Low | Low | Low | * Communicate via texts/email or communication services |
| Member gets sick | Low | Low | Medium | * Help with members work |
| Building the wrong product | High | Low | High | * Frequent meetings with client/advisor |
| Going down the wrong path | High | Medium | High | * Frequent meetings with client/advisor |
| Bad estimates | High | High | Medium | * Set short term goals / Frequent retrospectives |

## Software Engineering Tools

This project will require several software tools. The following list provides them and provides an explanation of why they are needed.

APIs/Languages:

* C# - Language used for Unity Framework
* Unity framework - API to enable us to make our Windows application and easily draw graphics

IDEs:

* Microsoft Visual Studio - enables us to work with C# and make Windows applications
* Unity Editor - enables us to create graphics and our web-based application
* MonoDevelop - Provides an additional editor within Unity to write C# code

Issue Tracking:

* GitHub - Provides us with an easy way to show issues with our project and it’s the same site our repository will be in

Version Control:

* GitHub - Our repository hub that we will store each versions of our software. Our source code and version control will be managed here

Testing:

* Microsoft Visual Studio - Has great debugging tools and built-in Unit testing project suites.

Other Tools:

* Trello - Help us organize our workload and display the tasks that need to be done
* Google Drive - Provides us with an easy way to store documents/ files and collaborate on them at the same time
* Microsoft Project - Help us organize our project with deadlines and tasks that need to be done
* Microsoft OneDrive - Will allow us to produce word documents natively, we will be moving away from Google Docs and to OneDrive during iteration 2 to make document submission easier.

# Requirements

Hardware and Software Requirements

*In this section we will discuss the hardware and software required to run our application; specifically we will detail the space, processor, graphics capabilities required as well as the required disk space. Because our application is being developed for both the web and windows, there will be some differences in requirements between the two implementations. First we discuss the Windows/Excel application then we discuss the web application.*

Our application will be reliant on the unity engine to do many of the drawing tasks therefore the requirements of our application will mirror those of the unity engine. Beyond the unity engine our application will require more memory depending on the amount of data being parsed.

The windows application will require that:

-The user's system will need to be running Windows XP+

-The user's graphics card or chip will need to be capable of running DX9.

-The user's CPU must support SSE2 instruction set

-The user's hard drive will require at least 50 MB of free space

-The user will require at least 1 GB of ram and that amount will scale up with input data

The web application will require that:

-The users system will need to be running Windows XP+, Mac OS X 10.6+, Unbuntu 10.10+

-The user's graphics card or chip will need to be capable of running DX9.

-The user's CPU must support SSE2 instruction set

-The user will require at least 1 GB of ram and that amount will scale up with input data

-The user will be required to run Internet Explorer, Chrome, Firefox or Safari

So long as these basic requirements are met the user will be able to run the program. We recommend however that the processor be clocked at minimum 2.0 GHZ so that animation and drawing occur smoothly.

Our target platform and development platform will be the same version of Windows 7, as such we predict that the application will run correctly once deployed. We plan to run the application on CWU computers periodically to ensure that no platform based issues occur.

The web implementation will require a small amount of bandwidth, there will be no client/sever interaction beyond loading the application. We expect that the total data exchanged will be approximately 1-2 MB per user.

## Functional/Nonfunctional Requirements

*In this section we will cover functional and nonfunctional requirements, provide use cases to describe these requirements and prioritize each requirement appropriately.*

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| Load Dataset  ID: 0 | Functional | 0 - v high | The program must be able to load a user’s dataset. This may be achieved either by selecting an exported excel file or by pasting the dataset into the program itself |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| Parse Dataset  ID: 1 | Functional | 0 - v high | The program must be able to parse the dataset into a structure that Unity can work with. This includes error-checking and informing the user of non-parsable data |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| Provide Alternative Data Views  ID: 2 | Functional | 0 - v high | Trends in data are only recognizable when the user has several different views to visualize their data. Provide several views for any given dataset. |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| Change Current View  ID: 3 | Functional | 0 - v high | Users should be able to select which view that they wish to apply to their dataset. |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| Change Dataset  ID: 4 | Functional | 0 - v high | Users should be able to load a new dataset on the fly. |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| Local Deployment  ID: 5 | Functional | 0 - v high | The program must be able to run standalone in Windows. |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| Web Deployment  ID: 6 | Functional | 0 - v high | The program must function on the web and behave the same as the standalone version. |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| Animation  ID: 7 | Non-Functional | 1 - high | Displayed data should appear and interact with the user in a pleasant way |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| Interactivity  ID: 8 | Functional | 1 - high | The user should be able to select lines and be given an indication as to what lines they have selected + any other useful info(TBD) |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| GUI  ID: 9 | Nonfunctional | 0 - high | Graphical User Interface should allow intuitive interaction between the program and the user, providing access to different areas of the program. |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| Excel Plugin  ID: 10 | Functional | 3 - med | The package should include an Excel plugin(s) that display graph views on highlighted data in MS Excel. |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement Name | Type | Priority | Description |
| Easter Eggs  ID: 11 | Nonfunctional | 5 - low | The program should have some fun hidden stuff. |

Use Cases

Requirement: Load Dataset

* User opens application
* User selects “Load Dataset”
* User chooses a .csv file with appropriate formatting
* Dataset is Loaded and displayed by default view

Requirement: Parse Dataset

* User opens application
* User selects “Load Dataset”
* User chooses a .csv file with appropriate formatting
  + Program reads incoming data stream
    - Program gives indication that data was loaded successfully
    - Program gives indication that data was not loaded successfully

Requirement: Provide Alternative Data Views

* User successfully loads a dataset
* User views default dataset presentation window
* User is presented with option to view dataset in alternative modes

Requirement: Change Current View

* User successfully loads a dataset
* User views default dataset presentation window
* User is presented with option to view dataset in alternative modes
* User selects alternative view mode
* Alternative view mode is displayed
* User is presented with option to view dataset in alternative modes

Requirement: Change Dataset

* User is viewing a dataset
* User selects “Change Dataset”
* User chooses a .csv file with appropriate formatting
* Dataset is Loaded and displayed by default view

Requirement: Local Deployment

* User obtains standalone copy of program
* User runs standalone program in Windows at any time

Requirement: Web Deployment

* User browses to Data Visualization website in WebGL enabled browser
* User is presented with a fully functional WebGL enabled version of the program

Requirement: Animation

* User successfully loads a dataset into the program
* Visual representation is drawn in an aesthetically pleasing manner
* User selects a point on the data plot
* Selected point is highlighted and redrawn to emphasize its location

Requirement: Interactivity

* User successfully loads a dataset into the program
* Visual representation of dataset is displayed
* User selects a point on the data plot
* User is presented with visual feedback that point was selected
  + User may choose to zoom on point
  + User may choose to highlight related points
  + User may choose to locate selected point in table of original dataset

Requirement: GUI

* User launches application
* User is presented with buttons indicating various actions they can take

### Use case diagram for user:

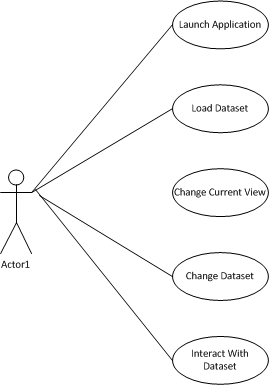


Figure 2 Use case diagram for user

## Feasibility

*This section provides a high level description of the essential and desired systems as well as an analysis of their feasibility (essential/enhanced).*

Our project’s base requirements are entirely feasible. We are confident in our ability to produce a program with base functionality within the proposed deadlines. To complete the project we are learning the C#, Unity, WebGL (minimal), HTML (minimal), and portions of the Office 365 plugins SDK. Although learning each of these items could take more time than we have allocated, we have limited ourselves to learning project-essential components and will be dividing our time to maximize project efficiency.

The basic version of our application will provide a GUI that allows a user to import specifically formatted multidimensional data from Excel (or another spread sheeting program) and display it using one of several 2D graphing techniques developed by Dr. Kovalerchuk. This program will run as a Windows standalone application as a WebGL application. Once a dataset has been successfully parsed, the user will be shown a “default” view, displaying their data as a 2D graph. While viewing their data, the GUI will provide the user with the option to change what style of graph that their data is presented in. This will allow the user to “thumb through” different graphing methods until they find a graph that appeals to them. The GUI will also provide the user with the option to change datasets, allowing them to load in a new dataset at any time. Developing in Unity will allow us to centralize the bulk of our project within a single codebase. Code written for the standalone program will be portable to the web, allowing us to quickly develop and test what works and what does not on both the web and standalone versions of the program.

The extended version of our program provides the user some interactivity with their dataset. For example, once a dataset has been successfully loaded and is being displayed, the user will be able to select individual points on the graph. Selected points may be zoomed upon, cleaned up (isolated and redrawn), referenced (highlighting corresponding coordinates on the imported table) and featured with color and/or animation. Because some datasets provide messy graphs, allowing the user to isolate and reference individual items on the graph will provide a valuable set of ‘extended’ features.

We hope to further extend this project by providing a plugin for Excel Online (through Office 365). The proposed plugin would allow a user to highlight their multidimensional data and choose an option to “push” this data to our WebGL visualization application. This set of features has been identified by our team as having the potential to be the most time-consuming component of our application and thus has been prioritized lower than the application + feature set previously described.

# Design

The following sections provide detail information on the design of our project. These sections include the architectural design describing each part of the system and their interaction, the user interface and the UIs features, software and pattern designs as well as a file structure of our applications.

Architectural Design

*In the following paragraphs we present a high-level overview of the major components of our system. Furthermore, we will describe the interactions between each component and what each component does.The figure below demonstrates a graphical representation of our system making it easier to visual their interactions.*

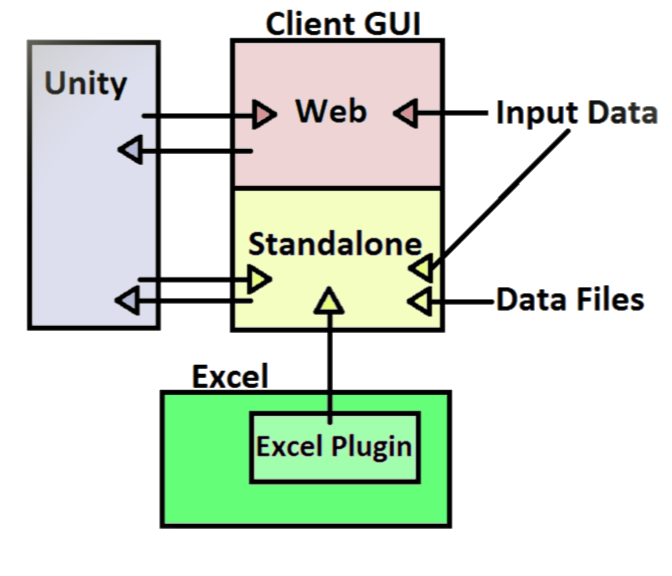


Figure 3 High level design

Figure 3 demonstrates the three major components of our system. The main component is the Unity Engine; Unity will process the input data and provide a graphical representation of the data through different types of graphs. It will also port our application into a web application and a Windows standalone application. This will allow us to develop under one framework and be able to easily port our application into other platforms. Within Unity we will use OpenGl to increase the performance of drawing the graphs as it is faster and more efficient than using Unity's objects.

The second component is our client GUI. There will be two GUIs, a web and a Windows standalone application. The user will only be able to interact with our system through these GUIs. The web and standalone application have a few differences. The web will only be able to accept data the user inputs by typing it in a text box or pasting data. The Windows application will be able to handle input data, reading from files such as excel and csv.

The last component in our system is the excel plugin within excel. The plugin will allow users of excel to visual their data in our standalone application. The users will have to add the plugin to excel by a simple option in excel to add plugins. Once done, the user will be able to use the plugin in excel. The plugin will simply use the user selected data in the excel sheet; this data will be pushed to the Windows application where the user will be able to visual their data in different types of graphs. In addition, the plugin will also notify the Windows application when data in excel is changed therefore update the graphs accordingly.

## User Interface Design

*The user Interface section describes how we organized our Interface and provides screenshots of the current iteration of the GUI. We also explain the design philosophy we used to accommodate the user requirements of this system.*

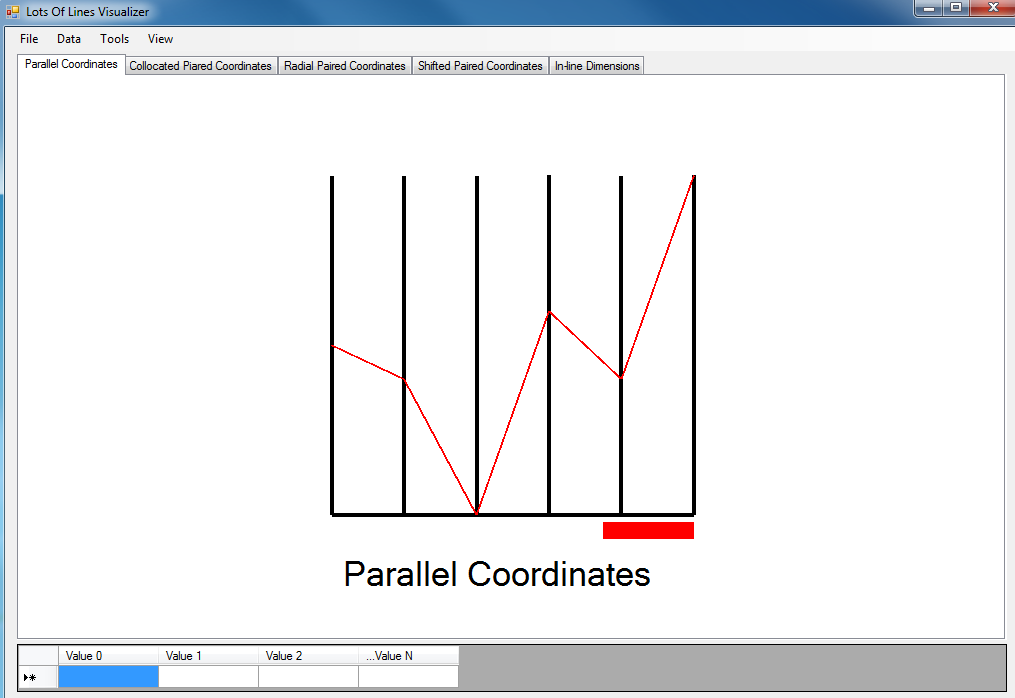


Figure 4 Parallel Coordinates



Figure 5 Data table

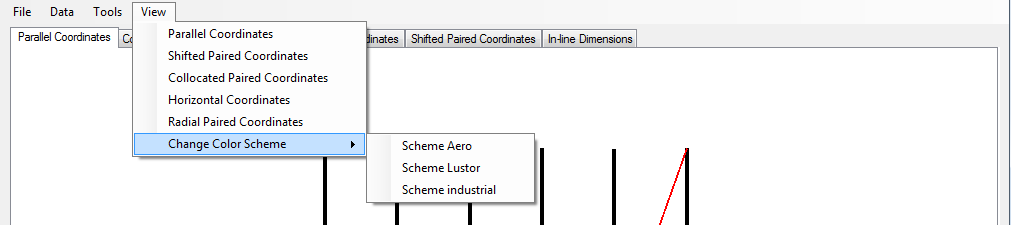


Figure 6 Menu

Figure 4 depicts how the program will appear when launched, without initial data the plots will appear blank. The table in figure 5 will also appear blank by default. Data will be displayed in the table and columns will scale to accommodate the correct number of values in a vector. Each row in the table will correspond with an input vector.

Currently we used a tabbed view to separate the five types of plots. In the Unity implementation of the GUI we plan to use dynamic cascading windows so that we can compare plots side-by-side.

Figure 6 depicts how the application's menu is organized. The menu provides options that allow the user to load and save data, find critical points, zoom, choose the visual representation technique calculated/shown, and change the coloration of the graph.

Our goal was to provide an interface that would be familiar to users of graphing software such as Excel or online graphing tools. We display the numerical data on the screen while displaying the plots so that users can test changes and see how single vectors affect the entire plot instantaneously. The user should be able to easily pull data from a file, see how it is displayed on the various plots, look for trends on those plots using the tools from the menu and change the input data in the data table at the bottom of the screen.

# Software Design

*The following section will detail our software design. It will explain, in detail, each portion of the previously depicted high level design through the use of static structure diagrams and accompanying explanations.*

Our software solution requires a design the will be deployed as a Windows standalone application and as an embedded web application. To centralize our project and facilitate coherence, we have chosen to develop our application utilizing the Unity framework. By developing our application in Unity using C#, we will be able to implement our design once and then deploy our application to its respective web and standalone components. In doing so, we are saving ourselves the trouble of managing two entirely different codebases. When we notice an issue in one deployment of our program, say the web component, we can easily return to our single codebase, fix the issue, and then test both the web and standalone components to make sure that both the issue is fixed and that the respective fix did not break the standalone component. As both deployments of our program will have identical functionality(except for Excel communication support), we will overview our software design for the web and Windows standalone deployment simultaneously. When it comes time to detail data input, our deployments will behave differently, which will be explained in the last section.

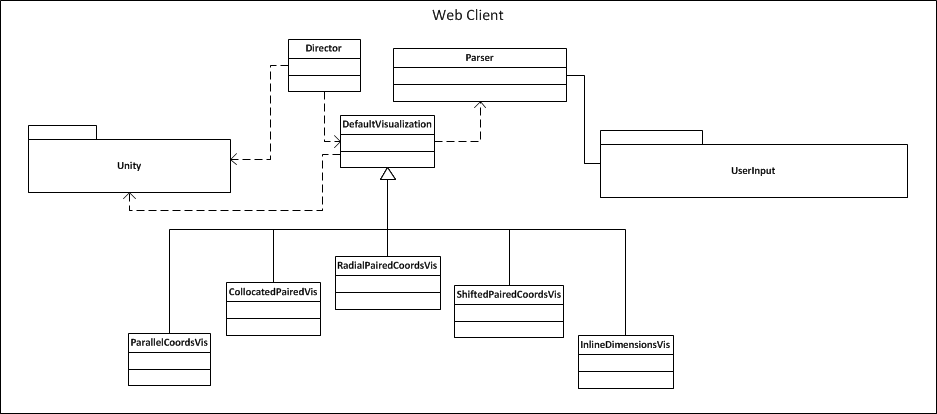


Figure 7 SS diagram for web client

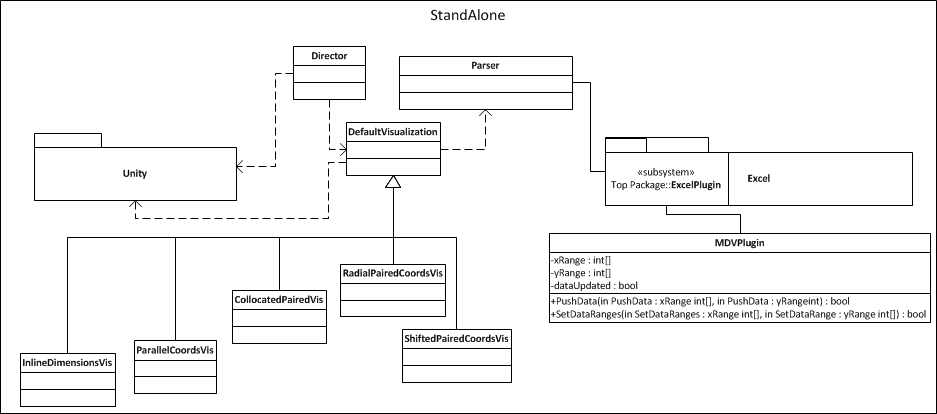


Figure 8 SS diagram for Windows standalone

The above static structure diagrams depict how our program will operate. At their core, our deployments will accept multidimensional data through a Parser object. This parser is responsible for verifying the integrity of the given data and breaking it down into a format that will be read through the shared implementation of our visualizations.

DefaultVisualization is an abstract class that defines what functions a class inheriting from DefaultVisualization must implement, such as drawing functions and special data segmentation. As presented in our requirements, we will be providing 5 visualizations. Each of these visualizations define the unique way that the parsed data is to be drawn to the screen. Because of the fact that Unity's Line Renderer functions are not optimized for drawing a large number of lines each frame, our data visualizations will drop down to a lower level and leverage OpenGL types and functions.

A Director class will manage world space operations such as vertex and line selection in addition to tracking mouse movement. The director class is responsible for providing the implementation to isolate selected vertices from a Visualization, and inform the visualization as to which data points are to be updated or segregated into special segments for alternate operations during the drawing pass.

The Director and each of our visualizations will interact with the Unity Assemblies and adhere to Unity's format for frame updates and user input to allow the application to be compiled and deployed to the web and a Windows standalone application.

Our two products will diverge when it comes to where the Parser class receives data from. In our web application, data may be pasted directly into our application window or be opened by selecting an Excel or CSV file to be read. Implementation for the file reader and pasted-text parser will be straightforward. The Windows standalone application will have the capacity to accept data in the same manners that the web version does, but includes an additional Excel plugin feature.

We will be developing an Excel plugin that will be responsible for watching tagged cells and pushing them to our Windows standalone application process. The plugin will be responsible for monitoring an Excel spreadsheet and only pushing updates to the standalone application when the spreadsheet data changes. It has been requested that this functionality be extended to the web version of our software without requiring a browser plugin, but additional research is required to determine if this safely possible.

Design Patterns

*The following section describes the use of design patterns in our system. We use several design patterns. Our project makes good use of model view controller, the observer pattern, the command pattern and the lastly the Façade pattern.*

The model view controller pattern is demonstrated with our Excel data, the data parser and the different visualizations. The model is the different graphs that our application can produce. Appendix a demonstrates these different classes that are children of Default Visualization class. The controller is demonstrated by the data the user selects to visualize through the GUI. These include, selecting some data, loading a data file and parsing user input. This will be handled by the parser class. The view in this design pattern are the different graphs the application will display of the data; consequently, the view will provide the user with different visualization methods.

The observer pattern is also used in our project. Our director class acts as the subject as well as the parser class. Both of these classes notify the Default Visualization class of changes in the data or data selected. As shown in in appendix a, the parser class has a NotifyVisualization() method that will notify the graphing classes of change of data and to update their visualization accordingly. Hence, the Default Visualization class acts as an observer of both the parser and director classes. The children of Default Visualization are then notified of the changed therefore updating and redrawing the graphs.

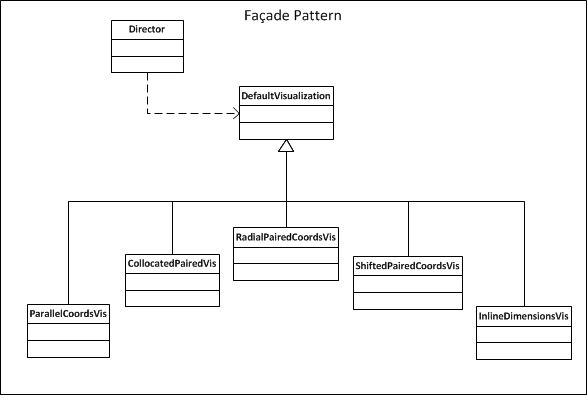


Figure 9 MVD facade pattern

The Façade pattern is utilized in our system when it comes to interaction between the Director and any visualization. The Director will need to interact with any of the visualization types at any given moment, thus it will interact with methods defined in DefaultVisualization to complete tasks in any of the other visualizations.

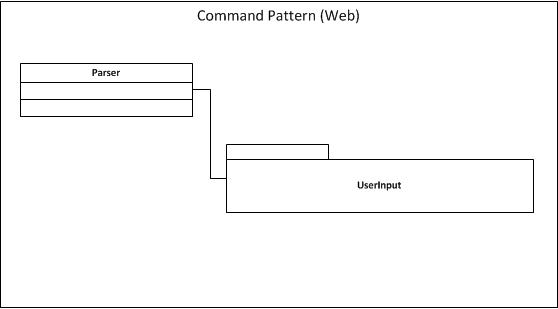


Figure 10 MVD command pattern(web)

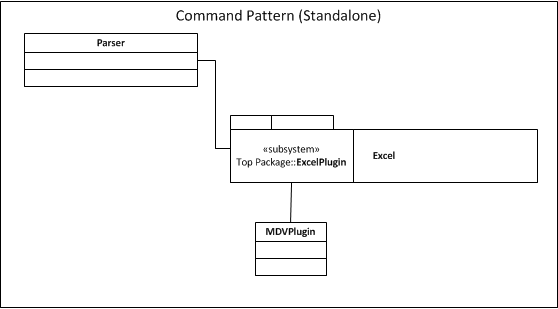


Figure 11 MVD command pattern (standalone)

Finally, the command pattern is utilized(most obviously) in the interaction between the Excel plugin and the Parser. The Excel plugin will be responsible for pushing data to the Parser, setting off a series of updates in the standalone program. In the web version, a UserInput controller will be responsible for kicking off this command by either loading a file or pasting in data.

Website Design

Our website implementation will happen through two potential ways depending on how development progresses. The first option we have in deploying our application is through the Unity's web plugin. Using this plugin, our application would look exactly identical as the desktop version except it would be within the web browser. The only downside to this is that the user is required to download a plugin, thus potentially causing headaches for users, and possible incompatibilities. The second possibility we have in deploying our application to the web is using WebGL. Unity3d is slated to release an update in early 2015 that will allow us to release our application in WebGL. However if this update does not come in time, we will either have to rewrite our application in WebGL manually or just use the Unity web browser plugin. If we manually rebuild the application in WebGL, we will use bootstrap to quickly create and iterate our interface for the application. The web application will have all the same functionality as the desktop app, we will demonstrate the flow of the application with a site map.

The homepage of our application will contain our application and have links to contact our development team and to navigate to our applications documentation. If users want to manually input data they can do this from the main application window. Below is a site map:

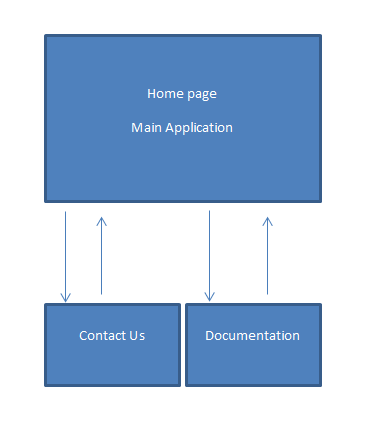


Figure 12 Site map

# File Structure

*The following section details the file structure of our system.*

Our system interacts with files directly by importing data from a specified file to be read and parsed, and by saving out modified data as a comma-separated-values file. It is neither required nor expected that users will input data from a file or save their data to a file on every occasion that our program is run; however, by providing the ability for the user to perform these operations is essential to increasing the usability and usefulness of our system.

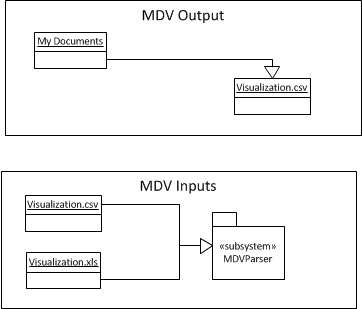


Figure 13 MVD Input/Outputs

# Quality Assurance

In this section we will outline the documentation and coding standards that our team has agreed upon for the purpose of quality assurance (QA). Adhering to these standards will enhance our ability to create a quality product and facilitate a project environment that facilitates a greater understanding of each team member’s actions.

## Document standards

In the interest of providing high quality documentation, we have agreed on several standards that must be followed by each member of the team:

1. The master iteration report document will be stored on a shared Google Drive, giving each member access to the most recent version of the document.
2. Each team member will be responsible for creating a rough draft of at least two sections of the iteration report.
3. Each document section will be peer reviewed by one or more team member before being considered “complete”.
4. The documentation lead will conduct a final review of the document layout to ensure it flows in an orderly and logical manner, rearranging sections as needed.

Coding standards

*This section provides information about the coding standards we plan to follow.*

In the interest of maintaining clean, organized and readable code, the team has agreed on several standards and will be adhering to them throughout production of our project. Current coding standards include Camel Cased variables (e.g.: int currentPopulation vs int currentpopulation), capital function names (e.g.: MyFunc(a,b) vs myFunc(a,b), where functions are easily distinguishable from variables), same-line brace openings (e.g.: beginning a function’s opening curly brace on the same line as it’s header), and a massive amount of clean commenting (e.g.: detailed multi line commenting a function or task prior to its creation).

Change Control Process

*In this section we discuss how we will handle a change in the requirements without significantly interrupting our development process.*

In the event that a change of control is required due to a failure in our design or an update in requirements, we will update a series of systems to reflect the change and redistribute our workload. The systems that will need to be updated are Trello and Github. Additional tools that will need to be updated are our burndown and Gantt charts. Adding any additional info to the issues section of our Github project is required. These issues will be immediately readable and reviewable by the whole team. Trello will need to be updated with new boards to reflect the change in control process, some boards in Trello may need to be re-written or shelved entirely. Updating our burndown and Gantt charts will be critical in the event of a change of control process due to the fact that our team will be adhering to the timetable on these charts.

Testing Process

*In this section we will describe our testing plan and processes. We will address the different levels of testing that we will implement and discuss the importance of thorough testing within our project.*

Testing will be conducted at all steps of the development, Unit tests will be provided to insure that the mathematical functions are running correctly. There are a multitude of functions which could produce incorrect output; therefore, testing will be done incrementally so that each function is tested at its time of creation and when other functions call it. In this way, a collection of properly tested functions, which we can rely on to be correct, will emerge.

This method can be contrasted to testing at the end. If we test after all the functions have been implemented then it is impossible to isolate bugs. If an error exists it may be occurring several levels deep in function calls or in multiple functions.

Since we are working with Excel documents, we will do testing for various types and versions of Excel documents. We want to ensure that a document from excel 97 doesn’t cause our application to crash or that a new version of an Excel documents provides different graphs. Ultimately, we will do extensive testing with different versions of Excel as input for our application and specifically test corrupted Excel documents to make sure they don’t cause fatal errors in our application.

In addition to iterative unit tests we will test the graph visualization against simple two dimensional representations. To do this we will examine the shape of sections of data on our visualization and compare them to the shape of isolated data which corresponds with the observed section. This check will ensure that the correct visualization is being displayed.

We will do extensive user interface testing; we will run through every part of the interface within the team, and we will have users outside the team try the application, and observe their interactions with the applications. We will make sure that everything within the application is logical and every interaction is obvious to a user that has never used the application before. We will do our best to implement the four pillars of design. We will clearly define the user interface requirements, and try to implement some ethnographic observation and see exactly what our users need.

We will document well our entire user interface and all the processes needed to fully navigate the application. Unity has built in user interface tools and we will extensively use it to build our interface. The fourth pillar of design will be accomplished by our development team using and testing every part of the application; furthermore, we will attempt to create a group outside of our team to use and test out the application so we can fully understand its usability.

More generally we will apply the techniques of Acceptance testing, Integration testing and Installation testing. Acceptance testing will be implemented through feedback from the client, and by testing of fulfillment of the requirements.

Specifically we will do integration testing to verify data passed between the parser and the excel plugin. We will test for data integrity and for a successful transfer of data. We will do usability testing with our client to ensure that all the requested features are available and perform their specific tasks. We will do system testing to guarantee that functionality is identical between the web and standalone version. We will do unit testing on each of the transformation techniques.

# Conclusion

## Summary

The Lots Of Lines application will provide users with a tool for visualizing N-dimensional Data. N-dimensional Data is difficult to visualize in a way that patterns are easily extractable. The Lots Of Lines application will handle the problem by providing five transformations which migrate the complex patterns of n-dimensional data into two dimensional space. Each transformation will provide a different perspective from which the data can be viewed. Similar to how looking at a picture from a different perspective might provide a new understanding; by looking at all five visualizations for a set of data the user might find a new understanding of trends and patterns that exist.

## Retrospective

Our team has met with full attendance and participation for every planned event. We have met with our project advisor weekly (excluding one week where we experienced a communication issue) and have verified the base requirements for our project thoroughly. Additionally, our team morale is high! An issue that we have encountered is identifying quantifiable progress. Although we have accomplished a fair amount of work, most of our progress has been logistic and hard to quantify. We have decided to continue holding weekly meetings with our advisor to strengthen our project’s direction and progress. We have identified the most and least imperative project features and have prioritized them accordingly. Our team seems to work well together, sharing a drive to produce a quality product and has experienced only slight communication issues. To help improve our communication and progress, we will participate in Scrum meetings every other day, with an emphasis on what progress we have made and what help we might need. In order to improve our communication away from meetings, we have decided upon Canvas as a centralized medium for communication and have committed to check Canvas multiple times per day.

*Iteration 2 Retrospective*

During the second iteration we continued with full attendance and dedication towards our project. We kept meeting with our advisor as previously and have done much of the work required to start developing our applications. We have developed several prototypes to provide us as well as the our client of how our applications will work and what they are capable of. Working through our second iteration we learned much from the problem we are trying to solve as well as each other.We learned that everyone is well devoted to finishing and making this project successful; we also work well under pressure and can produce many ideas to solve problems. What we believe we need to improve upon is reanalyzing our short term goals. We may need to separate our goals into smaller goals to make them more feasible and perhaps get a better understanding of the goals. To address this, we plan on breaking down our goals further; additionally, we plan on analyzing them more deeply to be able to understand them and work with them.

## Status Report

Our team has already begun logistics for our project, including:

* Creating and utilizing a Trello board
* Downloading, reviewing and discussing Python code provided by Dr. K
* Creating project files, uploading them and sharing them out through a GitHub repo.
* Selected development tools/kits and began familiarizing ourselves with them.

Our project has a solid start, capable members and good pool of resources draw from. Given the scope and complexity of the project, we feel ahead of schedule at the conclusion of iteration 1.

*Iteration 2 Status Report*

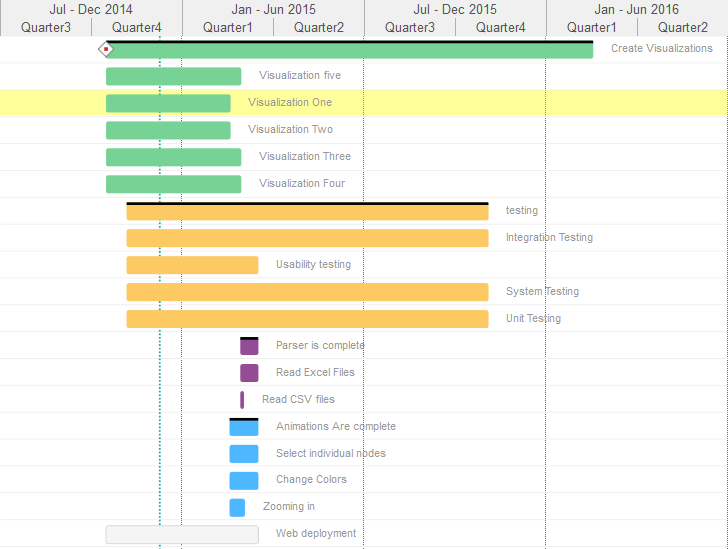
Our team has made headway in creating several prototypes to test the direction that we want to take our project in. We have:

* Successfully created a data parsing prototype that will accept a dataset from a text input field and then draw a graph based on that dataset
* Moved our line rendering strictly to OpenGL, rather than Unity's built-in line rendering system to increase performance when rendering a large number of lines.
* Constructed a mock UI to help us visualize how our final program will be laid out.
* Created a test Excel plugin that can manipulate spreadsheet data using MSDN tools and docs.
* Centralized our project on GitHub and have been developing test products from our single codebase.

Our project has begun moving from the conceptualization stage to the actualization stage. We have developed several prototypes and have become familiar with our code base. We are looking forward to combining our findings and efforts into our final product.

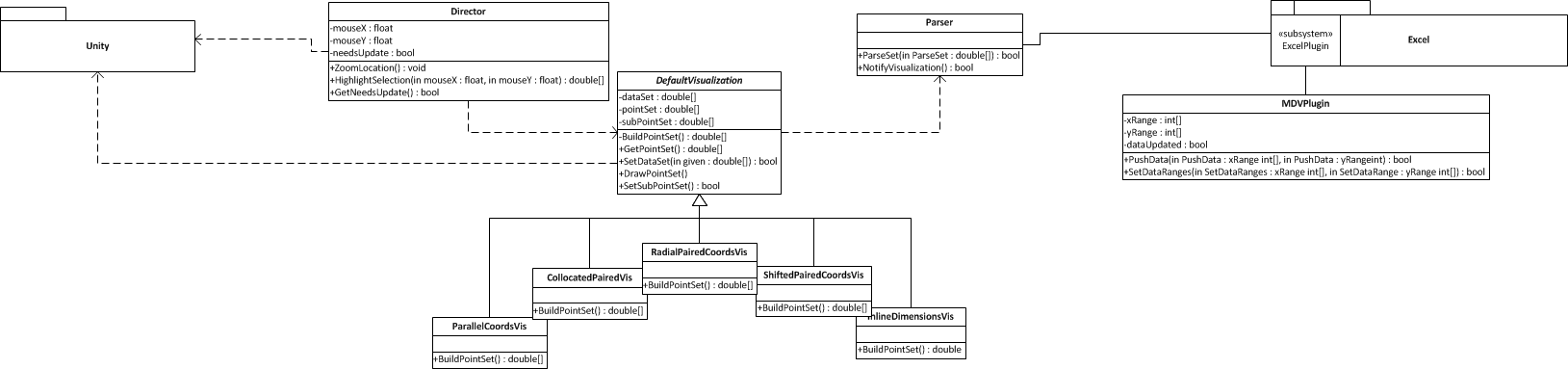
# Appendices

Appendix A



Appendix a | Gantt chart

Appendix B



Appendix b | Class Diagram

<https://trello.com/b/lbL8IU08/lossless-visualization-project>

<https://github.com/tommymathan/2dVisualizationUnity>