Diagram Rationale

By: Eric Bachmeier and Jennifer Xin

Use Case Diagram:
This diagram shows the user's interaction with our system and the cases that we provide the user with. As illustrated, a user can invoke any number of use cases such as *Choose File*, *Expand View*, and *Modify Graph*. The existence of use cases *Fix Errors* and *Ignore Errors* depend on the existence of *Verify Information* and hence are included by it. On the other hand, the use cases *Select Author* and *Select Type* are extended by *Modify Graph* because they are non-optional features that add to it.
Upon opening the application the user is presented with a loading screen before the main options are presented. The user then chooses the .csv file they wish to analyze data for and the file is verified. Errors are either ignored or fixed in this step pertaining to the .csv file loaded in. Once the user is happy with the data, they are brought to the analyze screen. Here, they can visualize the data in a personalized way with expanding tabs for information, graphs that can be filtered for certain data, and modifiable graphs by Author or Type of data.

UML Class Diagram:
This diagram offers a graphic representation of the classes in the system and the relationship among objects. Information such as attributes and method can also be derived.
The classes we chose to use for our application mainly pertain to the three steps of our application as the user goes through it; load, verify, and analyze. After being loaded in, the data is assembled into our distinct DTO objects from the .csv files. Once the data is verified, it can be analyzed by the user. The class diagrams shows how we modularized function in classes to multiple tasks to gather and display the data. First we populate the tree diagram with data and the combo boxes with years and graph names. After that is done we utilize our graph visualization classes to construct the plots to display.

Package Diagram:
This diagram illustrates the dependencies of packages (a group of related elements) in the system. We've chosen to organize classes into three large packages by functionality - *Load, Verify*, and *Analyze*. The latter two are further divided by their logical function into smaller packages. Categorized in this way, package diagrams allow for a big-picture view of the system that is easier to understand than a UML diagram.

Sequence Diagram:
Sequence Diagrams show the order (i.e. sequence) of interaction among the components of a system in order to perform a functionality, and the flow of messages from one object to another. We've illustrated the scenario of loading a CSV. The first call is *on_verify_btn_clicked(),* which is a method of *verify_cv* object. The next call to *csvdataassembl*er is for the method *AssembleData(). AssembleData* returns a value back to the class that called it, and the sequence of method calls continue on.