



Software Design Description

Version 3-2014-06-04

for

PlasmaGraph

Daniel E. Quintini Greco (# 73749)

Computer Science Undergraduate Program

Gerardo A. Navas Morales (# 69615)

Computer Science Undergraduate Program

**Polytechnic University of Puerto Rico
Electrical & Computer Engineering and Computer Science Department
Computer Science Senior Project**

Advisor: Prof. Luis A. Ortiz

Client: Prof. Angel Gonzales-Lizardo

Revision History

Date	Version	Reason For Changes
January 21, 2014	0-01-21-2014	First version of document. No diagrams, just Introduction. Using SPMP as template.
January 23, 2014	1-02-23-2014	Added diagrams made up to this date. Provided better structure and removed all mentions of SPMP, including Table of Contents.
May 22, 2014	1-05-22-2014	Changed format of document to better describe the important design features of PlasmaGraph. Added basic versions of the System Overview and System Architecture (mostly diagrams). Added complete versions of the Data and Human Interface Designs, as well as the Requirements Matrix. Added sample versions of Pseudocode and Activity Diagrams for Component Design to test the better code description vehicle between the two.
May 29, 2014	2-05-29-2014	Added formatting changes to title page. Corrected various textual errors in all sections. Added information to Sections 2, 3, and 6 to better explain functionality. Removed project backstory from Section 2, as it already exists in the SPMP and in the Introduction in an abbreviated form. Added Class Content and Relationship diagrams to Section 3. Added pseudocode to Section 5. Updated numbering scheme for Sections 3 and 5. Updated Table of Contents and List of Figures / Tables with new page numbers.
June 4, 2014	3-06-04-2014	Corrected grammatical errors. Added Help Manual image to Section 6.2. Replaced old Section 3.2.3 Sequence Diagrams with more accurate ones. Switched the positions of Sections 3.2.1 and 3.2.2 and moved some diagram components around to not overlap class boxes and lines, and to keep crossing lines perpendicular to each other. Added legend to Class Diagrams to describe attribute / method symbol meanings. Changed the layout of a few Class Diagrams.

Table of Contents

1. INTRODUCTION	1
1.1 Purpose.....	1
1.2 Scope.....	1
1.3 Overview	1
1.4. Reference Materials	1
1.5 Definitions and Acronyms	1
2. SYSTEM OVERVIEW	3
2.1. System Functionality.....	3
2.2. System Design Overview	3
3. SYSTEM ARCHITECTURE.....	4
3.1 Architectural Design	4
3.2 Decomposition Description.....	7
3.2.1. Class Relationships	7
3.2.2. Package Contents	12
3.2.3. Requirement Fulfillment	31
3.3 Design Rationale	42
4. DATA DESIGN	43
5. COMPONENT DESIGN	44
5.1. PlasmaGraph	44
5.1.1. main ().....	44
5.2. MainModel	44
5.2.1. importData ().....	44
5.2.2. importTemplate ()	45
5.2.3. saveTemplate ()	45
5.2.4. prepareDataLog ()	45
5.3. GraphModel.....	46
5.3.1. graph ().....	46
5.4. GraphView.....	46
5.4.1. graphUpdate ()	46
5.5. HeaderData.....	47
5.5.1. populateData (GraphPair p)	47
5.6. DataSet	47
5.6.1. toXYGraphDataset (GraphPair p).....	47
5.6.2. toGroupedXYGraphDataset (GraphPair p).....	48

5.7. MatlabProcessor	48
5.7.1. getHeaders (HeaderData hd).....	48
5.7.2. toDataSet (DataSet ds, GraphPair p, HeaderData hd)	49
5.8. Template	49
5.8.1. saveTemplate (String s).....	49
5.8.2. openTemplate (File f).....	50
5.9. Interpolator	52
5.9.1. interpolate ()	52
5.9.2. getInterpolation (XYSeries s)	52
5.10. ClusterScanning.....	53
5.10.1. scan (HeaderData hd, Template t, GraphPair p).....	53
6. HUMAN INTERFACE DESIGN	54
6.1 Overview of User Interface.....	54
6.2 Screen Images	55
6.3 Screen Objects and Actions.....	59
7. REQUIREMENTS MATRIX	61

List of Figures

Figure 1: PlasmaGraph Package Diagram	5
Figure 2: Main MVC Class Relationships	7
Figure 3: Data Set MVC Class Relationships	8
Figure 4: Tool MVC Class Relationships	8
Figure 5: Graph MVC Class Relationships	9
Figure 6: General MVC Class Relationships	9
Figure 7: Data Class Relationships	10
Figure 8: Data Reader Class Relationships	10
Figure 9: Template Class Relationships	10
Figure 10: Graph Class Relationships	11
Figure 11: Outlier Searching Class Relationships	11
Figure 12: Class Contents Diagram - Symbol Terminology Legend	12
Figure 13: PlasmaGraph Class Contents Diagram	12
Figure 14: MainController Class Contents Diagram	13
Figure 15: DataSetController Class Contents Diagram	14
Figure 16: ToolController Class Contents Diagram	15
Figure 17: GraphController Class Contents Diagram	15
Figure 18: MainModel Class Contents Diagram	16
Figure 19: DataSetModel Class Contents Diagram	16
Figure 20: ToolModel Class Contents Diagram	17
Figure 21: GraphModel Class Contents Diagram	17
Figure 22: Automated Test Class Contents Diagram	18
Figure 23: FileUtilities Class Contents Diagram	18
Figure 24: HeaderData Class Contents Diagram	19
Figure 25: HeaderComponent Class Contents Diagram	19
Figure 26: GraphPair Class Contents Diagram	19
Figure 27: DataSet Class Contents Diagram	20
Figure 28: FileProcessor Interface Contents Diagram	21
Figure 29: MatlabProcessor Class Contents Diagram	21
Figure 30: Graph Interface Contents Diagram	21
Figure 31: XYGraph Class Contents Diagram	22
Figure 32: Template Class Contents Diagram	23
Figure 33: DataConfidence Class Contents Diagram	24
Figure 34: Interpolator Class Contents Diagram	24
Figure 35: OutlierSearch Class Contents Diagram	24
Figure 36: OutlierDistance Interface Contents Diagram	25
Figure 37: CartesianDistance Class Contents Diagram	25
Figure 38: MahalanobisDistance Class Contents Diagram	25
Figure 39: ScanMethod Interface Contents Diagram	25
Figure 40: ClusterScanning Class Contents Diagram	26
Figure 41: Data Type Class Contents Diagrams	26
Figure 42: MainView Class Contents Diagram	27
Figure 43: DataSetView Class Contents Diagram	28
Figure 44: ToolView Class Contents Diagram	29
Figure 45: GraphView Class Contents Diagram	29

Figure 46: DatasetLogView Class Contents Diagram	30
Figure 47: Help Manual Class Contents Diagram	30
Figure 48: “Import Data” Sequence Diagram.....	31
Figure 49: “Validate Data” Sequence Diagram.....	32
Figure 50: “Choose Graph Options” Sequence Diagram.....	33
Figure 51: “Create Graph” Sequence Diagram.....	34
Figure 52: “Save Graph” Sequence Diagram	35
Figure 53: “Save Template” Sequence Diagram.....	36
Figure 54: “Import Template” Sequence Diagram.....	37
Figure 55: “Inspect Data” Sequence Diagram	38
Figure 56: “Display Help” Sequence Diagram	39
Figure 57: “Choose Location” Sequence Diagram	40
Figure 58: “Elicit Graph Options” Sequence Diagram.....	41
Figure 59: PlasmaGraph Graphical User Interface Windows.....	55
Figure 60: PlasmaGraph Settings Window Tabs.....	56
Figure 61: Open File / Template Window.....	57
Figure 62: Save Template Window	57
Figure 63: View Data Window.....	58
Figure 64: User Manual Window.....	58

List of Tables

Table 1: Document Definitions	1
Table 2: Document Acronyms.....	2
Table 3: Requirements Matrix.....	61

1. INTRODUCTION

1.1 Purpose

This document, the Software Design Description (SDD) [1], is designed to describe the architectural and system details of the PlasmaGraph product to all whom are interested in understanding the design decisions made in the creation of this product; specifically, this document is made for the students working at the Polytechnic University of Puerto Rico's Plasma Laboratory and any future contributors to the PlasmaGraph project. It explains what the program does and why it does it in order to provide an accurate synopsis of how the functional and non-functional requirements are achieved. Diagrams and pseudo-code will be both used in order to appropriately and concisely detail how PlasmaGraph will satisfy the requirements listed in the System Requirement Specification (SRS) [2] document. In doing so, readers will obtain a close-to-exact idea of how the program will function without having to look at the code itself.

1.2 Scope

The project is a graphing solution called PlasmaGraph. PlasmaGraph is a data-graphing program specifically made to produce graphs from the PUPR Plasma Laboratory's Matlab-encoded data files. It allows the user to select the Matlab-encoded data file, select the graphs columns and other visual settings, generate regressions based on one or more groups of data, and save the resulting graphs. PlasmaGraph will be used at the end of the experiment analysis pipeline of the Plasma Laboratory in order to visualize experiment data tendencies without resorting to tools that require programming knowledge, such as Matlab [1]. Therefore, this product would be indispensable to new members of the Plasma Laboratory work team and older members who want to quickly view and understand experiment data.

1.3 Overview

The SDD is divided into various sections, each emphasizing an aspect of PlasmaGraph.

- System Overview (Section 2): Provides information regarding PlasmaGraph's key features based on the SRS's Functional and Non-Functional Requirements and the methods by which PlasmaGraph will implement the features.
- System Architecture (Section 3): Describes PlasmaGraph's architectural structure and how it

supports the requirements, and explains the final structure's design rationale.

- Data Design (Section 4): Details the various PlasmaGraph code objects that support the requirements.
- Component Design (Section 5): Describes the processes detailed in Section 3's Sub-Section B, "Requirement Fulfillment", which implement PlasmaGraph's requirements.
- Human Interface Design (Section 6): Describes the components of the visual component of PlasmaGraph, the Graphical User Interface, and how it supports the program's requirements.
- Requirements Matrix (Section 7): Details what parts of the PlasmaGraph program contribute to the functional and non-functional requirements.

1.4. Reference Materials

- [1] Object Refinery Limited, "JFreeChart," Object Refinery Limited, 25 November 2013. [Online]. Available: <http://www.jfree.org/jfreechart/>. [Accessed 13 May 2014].
- [2] The MathWorks, Inc., "MATLAB – The Language of Technical Computing," [Online]. Available: http://www.mathworks.com/products/matlab/?s_tid=hp_fp_ml. [Accessed 12 May 2014].
- [3] Oracle Corporation, "What is Java and why do I need it?," 25 March 2014. [Online]. Available: http://www.java.com/en/download/faq/whatis_java.xml. [Accessed 13 May 2014].
- [4] Merriam-Webster, "Dictionary and Thesaurus – Merriam-Webster Online," [Online]. Available: <http://www.merriam-webster.com/>. [Accessed 12 May 2014].
- [5] B. Goines, S. Chacon and M. McCullough, "Git – About Version Control," [Online]. Available: <http://git-scm.com/book/en/Getting-Started-About-Version-Control>. [Accessed 13 May 2014].
- [6] PC.net, "Definition of IDE," [Online]. Available: <http://pc.net/glossary/definition/ide>. [Accessed 13 May 2014].
- [7] Polytechnic University of Puerto Rico, "Polytechnic University of Puerto Rico – Main Page," [Online]. Available: <http://www.pupr.edu/>. [Accessed 13 May 2014].
- [8] Institute of Electrical and Electronics Engineers, "IEEE Standard for Software Project Management Plans," 22 December 1998. [Online]. Available: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=741937&isnumber=16012>. [Accessed 8 May 2014].
- [9] Institute of Electrical and Electronics Engineers, "IEEE Recommended Practice for Software Requirements Specifications," 22 December 1998. [Online]. Available: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=720574&isnumber=15571>. [Accessed 13 May 2014].
- [10] Institute of Electrical and Electronics Engineers, "IEEE Recommended Practice for Software Design Descriptions," 22 December 1998. [Online]. Available: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=741934&isnumber=16019>. [Accessed 13 May 2014].
- [11] Institute of Electrical and Electronics Engineers, "IEEE Standard for Software Test Documentation," 22 December 1998. [Online]. Available: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=741968&isnumber=16010>. [Accessed 13 May 2014].
- [12] GitHub, Inc., "About – GitHub," [Online]. Available: <https://github.com/about>. [Accessed 12 May 2014].
- [13] "How to Write Doc Comments for the Javadoc Tool," 16 November 2012. [Online]. Available: <http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html>. [Accessed 13 May 2014].

- [15] Git SCM, "Git – GUI Clients," [Online]. Available: <http://git-scm.com/downloads/guis>. [Accessed 14 May 2014].
- [16] R. C. Martin, "UML Java Programmers," 10 December 2008. [Online]. Available: http://www.csd.uoc.gr/~hy252/references/UML_for_Java_Programmers-Book.pdf. [Accessed 24 04 2014].
- [17] t. t. w. cbwatcham, "JMatIO – Matlab's MAT-file I/O in JAVA," 04 December 2012. [Online]. Available: <http://sourceforge.net/projects/jmatio/>. [Accessed 14 May 2014].
- [18] Apache Software Foundation, "Math – Commons Math: The Apache Commons Mathematics Library," 02 November 2013. [Online]. Available: <http://commons.apache.org/proper/commons-math/index.html>. [Accessed 12 February 2014].
- [19] Oracle Corporation, "Java SE Application Design With MVC," March 2007. [Online]. Available: <http://www.oracle.com/technetwork/articles/javase/index-142890.html>. [Accessed 15 May 2014].
- [20] R. C. Martin, UML for Java Programmers, Upper Saddle River, NJ: Prentice Hall PTR, 2003.
- [21] S. Ramanujan, "Software Design Document (SDD) Template," 7 May 2014. [Online]. Available: http://cmsu2.ucmo.edu/public/classes/sam/Advanced%20Systems%20Project/sdd_template.pdf.

1.5 Definitions and Acronyms

Unless otherwise specified, all definitions and acronyms are specific to this project's scope.

Table 1: Document Definitions

Term	Definition
PlasmaGraph / Product	A Matlab-file error-checking graphing program made for the PUPR Plasma Laboratory.
Client	Professor Angel Gonzales-Lizardo; the person requesting the project.
Advisor	University-designated overseer of the project's progress.
JFreeChart [1]	A set of tools written in the "Java" programming language that create graphical representations of data provided to it.
Matlab [2]	Data-manipulation and graphing IDE made by The Mathworks, Inc.
Java [3]	Object-oriented, interpreted programming language known for its portability between multiple operating systems and general-purpose capabilities.
Operating System [4]	Software that controls the operation of a computer and directs the processing of programs (as by assigning storage space in memory and controlling input and output functions).
Project Team / Team	The creators and maintainers of the project and all of its end products. Composed of Gerardo A. Navas Morales and Daniel E. Quintini Greco.
Version Control System [5]	Manages how multiple users can access and change the same files without losing data. Also known as a Revision Control System or Source Control System
Integrated Development Environment [6]	Software development program that keeps track of all files related to a project and provides a central interface for writing source code, linking files together, and debugging the software.
Vetting work packages	The process of reviewing a work package for semantical errors, documentation requirements (if a document), or programming errors (if a program).
Program Documentation	All Javadoc files and the User Manual.
Project Documentation	The SRS, SDD, STD, and SPMP documents.
Polytechnic University of Puerto Rico [7]	University where the Plasma Laboratory is located, as well as where the group's members study.
Software Project Management Plan / SPMP [8]	This document; One of the four IEEE project documents being created as part of this project.

Term	Definition
Software Requirements Specification / SRS [9]	One of the four IEEE project documents being created as part of this project.
Software Design Descriptions / SDD [10]	One of the four IEEE project documents being created as part of this project.
Software Test Documentation / STD [11]	One of the four IEEE project documents being created as part of this project.
Graphical User Interface / GUI [4]	A computer program designed to allow a computer user to interact easily with the computer typically by making choices from menus or groups of icons.
Development	Creation of product code.
PNG / .png / JPG / JPEG / .jpg / .jpeg	Types of computer files that hold image data.
MAT / .mat	Matlab data file format.
Contributor	A GitHub user status; allows the user to make changes to the project.
Incomplete document	A document with one or more sections labelled [TBD].
Complete document	A document with no sections labelled [TBD].
Document verification	The process an Advisor performs on a document to check for errors.
GitHub [12]	File repository system that uses the Git Version Control System. [12]
Package	A group of classes written in the Java programming language.
Class	A Java file that serves to encapsulate functionality within it.
Method	A unit of action contained within a class in the Java programming language.
Attribute	A container of information used within a class.

Table 2: Document Acronyms

Term	Acronym
PUPR	Polytechnic University of Puerto Rico.
SPMP	Software Project Management Plan.
SRS	Software Requirement Specification.
SDD	Software Design Descriptions.
STD	Software Test Documentation.
GUI	Graphical User Interface.
VCS	Version Control System
IDE	Integrated Development Environment.

2. SYSTEM OVERVIEW

2.1. System Functionality

As explained in Section 1.2, PlasmaGraph is designed to provide graphing tools to students and professors working in the PUPR Plasma Laboratory besides the tools provided by Matlab. The main advantage of the program over Matlab is its GUI providing a simple interface for producing graphs. The capabilities of the PlasmaGraph program, as defined by the requirements presented by the client in this project's SRS document, are as follows:

1. Import Matlab Level-5-encoded data files.
2. Validate data files for uneven column sizes or invalid data points.
3. Provide a Graphical User Interface for the user to select options.
4. Create graphs based on user-provided data and options.
5. Save graphs made by the user.
6. Save and import graph's options for later use.
7. Allow user to view data manually.
8. Provide a help manual within the program to explain how to use the application.

2.2. System Design Overview

PlasmaGraph is designed with Object-Oriented principles in mind, and will use the Model-View-Controller (MVC) pattern to implement user interface management. The program will be divided into roughly three categories:

1. Gathering and Storing Data: Data will be pulled from files and placed within various objects (listed in Section 4).
2. Gathering and Storing Graph Settings: The GUI will be handled via MVC objects, which will manage the visual positioning of interface objects as well as the signals sent to and from the interface and will form a part of the user settings for the graph.
3. Creating Graphs: The data and settings obtained in phases 1 and 2 will then be used to create graphs.

The design of the program, including how these three categories connect with the SRS's requirements, will be discussed with further detail in Section 3.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

As listed in Section 2.2, PlasmaGraph has a number of features it must contain in order to be considered a full product. These features are based on the functional and non-functional requirements stipulated in PlasmaGraph's SRS document. To implement these features, the program will group their implementation into three categories:

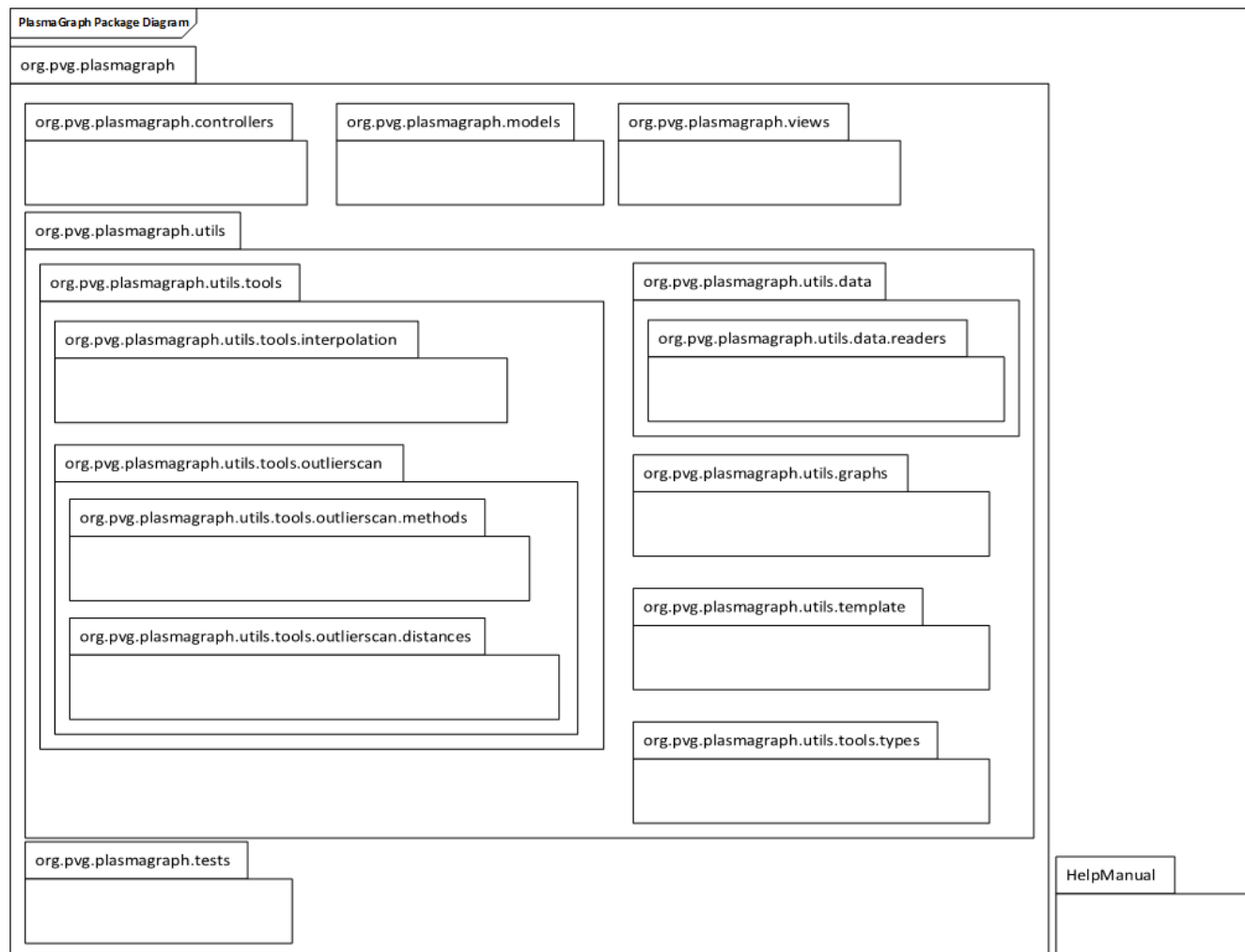
- Graphical User Interface.
- Data-reading and data-storing.
- Data-graphing and optional tools.

In following with the object-oriented design pattern, PlasmaGraph's features will be divided into objects whose specifications are located in various packages:

- All Class files related to the GUI are divided between the "org.pvg.plasmagraph.controllers" (Controller classes), "org.pvg.plasmagraph.models" (Model classes), and "org.pvg.plasmagraph.views" (View classes) packages.
- Data-Containing and Data-Reading Class files are located in the "org.pvg.plasmagraph.utils.data" and "org.pvg.plasmagraph.utils.data.readers" packages.
- Class files involving the Interpolation and Outlier Scanning tools are found in the "org.pvg.plasmagraph.utils.tools.interpolation" and "org.pvg.plasmagraph.utils.tools.outlierscan" packages or their sub-packages ("org.pvg.plasmagraph.utils.tools.outlierscan.distances" and "org.pvg.plasmagraph.utils.tools.outlierscan.methods").
- Automated test files are found in the "org.pvg.plasmagraph.tests" package.
- Files containing graph types and their procedures are found under "org.pvg.plasmagraph.utils.graphs".
- Template files are located under "org.pvg.plasmagraph.utils.template".
- Class files denoting unique types of options are found under "org.pvg.plasmagraph.utils.types".
- Files regarding the Help Manual are found in the "HelpManual" package.
- Most importantly, the file containing the program's initialization routine is found under "org.pvg.plasmagraph".

The following diagram depicts PlasmaGraph's package structure.

Figure 1: PlasmaGraph Package Diagram



Furthermore, most of these packages fall under the three categories in the following manner:

- Graphical User Interface.
 - `org.pvg.plasmagraph`
 - `org.pvg.plasmagraph.controllers`
 - `org.pvg.plasmagraph.models`
 - `org.pvg.plasmagraph.views`
 - `org.pvg.plasmagraph.utils.template`
 - `org.pvg.plasmagraph.utils.types`
 - `HelpManual`

- Data-reading and data-storing.
 - `org.pvg.plasmagraph.utils.data`
 - `org.pvg.plasmagraph.utils.data.readers`
- Data-graphing and optional tools.
 - `org.pvg.plasmagraph.utils.graphs`
 - `org.pvg.plasmagraph.utils.tools.interpolation`
 - `org.pvg.plasmagraph.utils.tools.outlierscan`
 - `org.pvg.plasmagraph.utils.tools.outlierscan.distances`
 - `org.pvg.plasmagraph.utils.tools.outlierscan.methods`

The “`org.pvg.plasmagraph.tests`” package is used for automated testing of other packages, and its purpose is described in better detail in the STD [11].

The table in Section 7 explains how each of the packages mentioned above help further each of the SRS requirements.

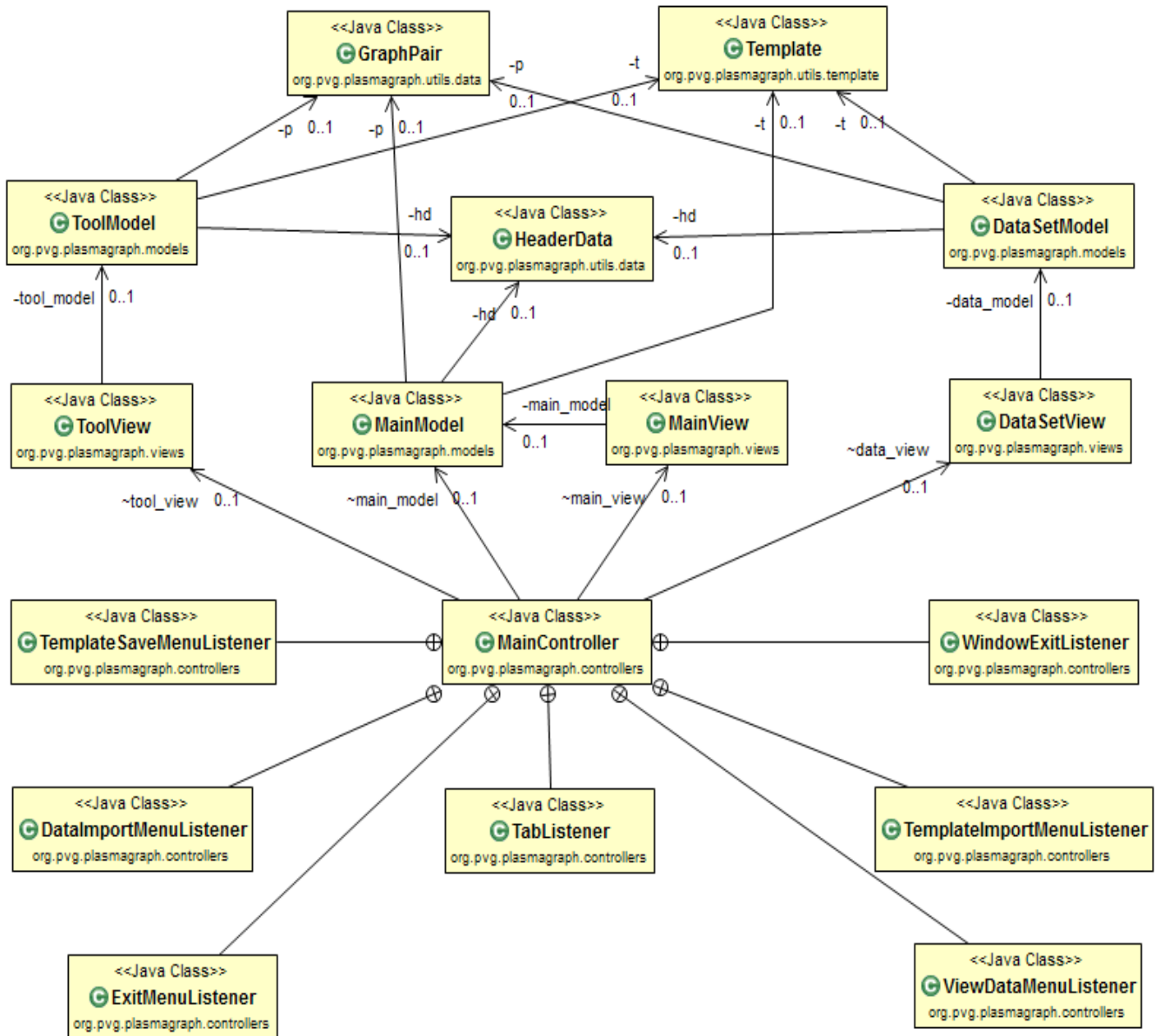
3.2 Decomposition Description

The following sections depict the code contained in the packages mentioned in Section 3.1 by way of describing the classes and how they're used in implementing PlasmaGraph's requirements.

3.2.1. Class Relationships

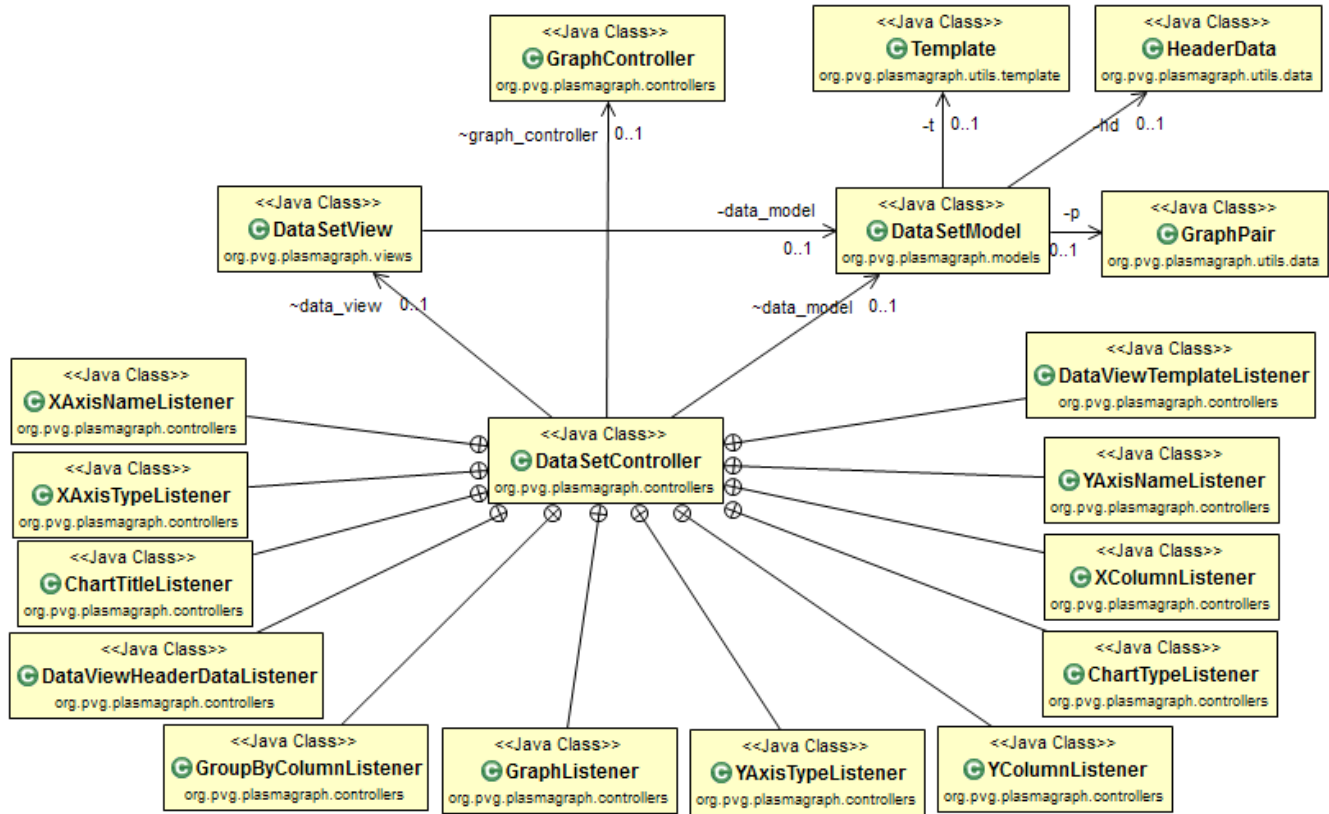
3.2.1.1. MVC – Main MVC Class Relationships

Figure 2: Main MVC Class Relationships



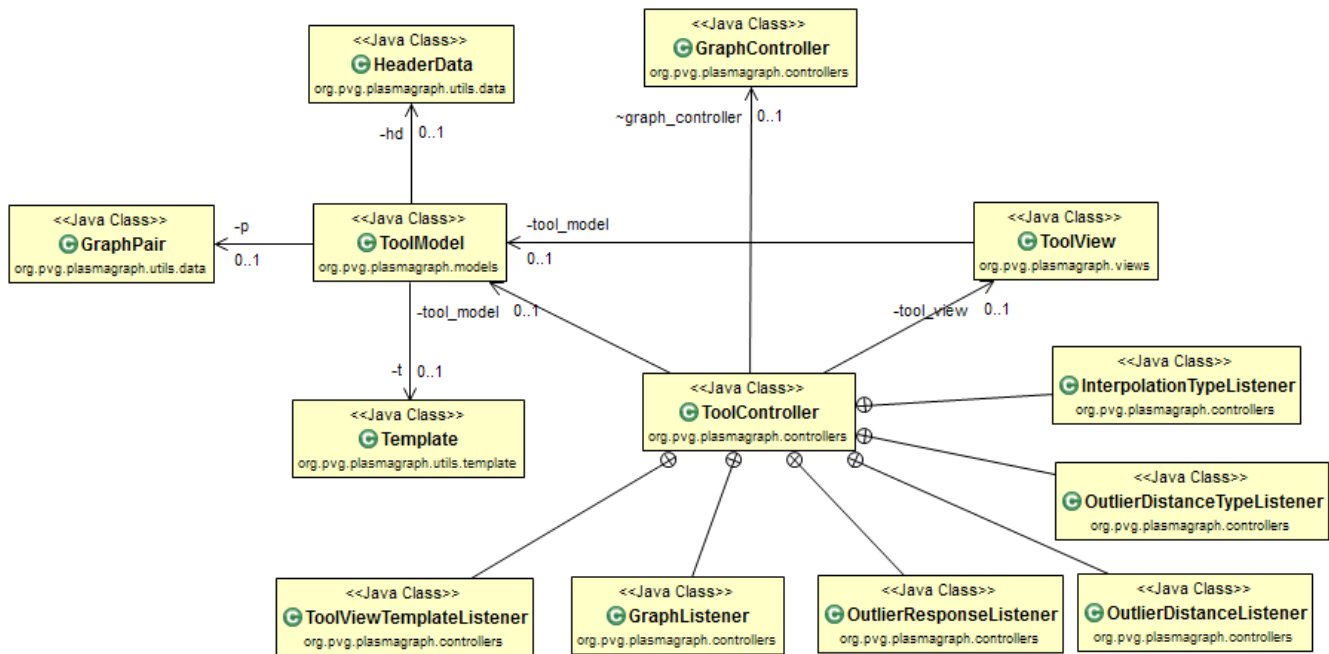
3.2.1.2. MVC – Data Set MVC Class Relationships

Figure 3: Data Set MVC Class Relationships



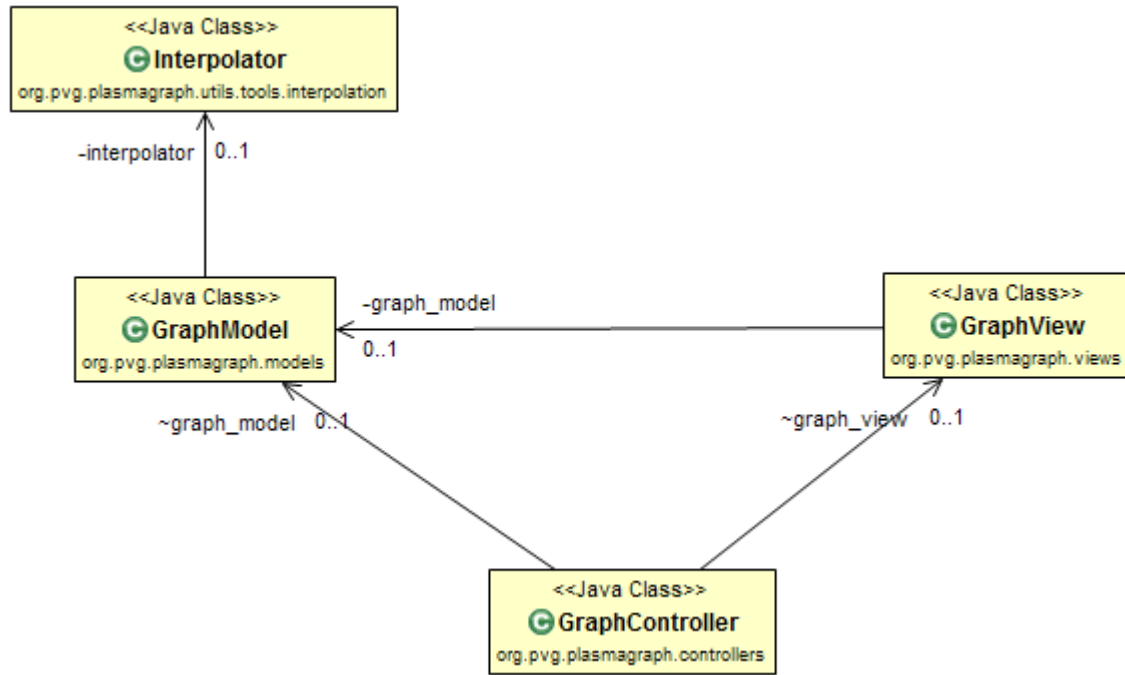
3.2.1.3. MVC – Tool MVC Class Relationships

Figure 4: Tool MVC Class Relationships



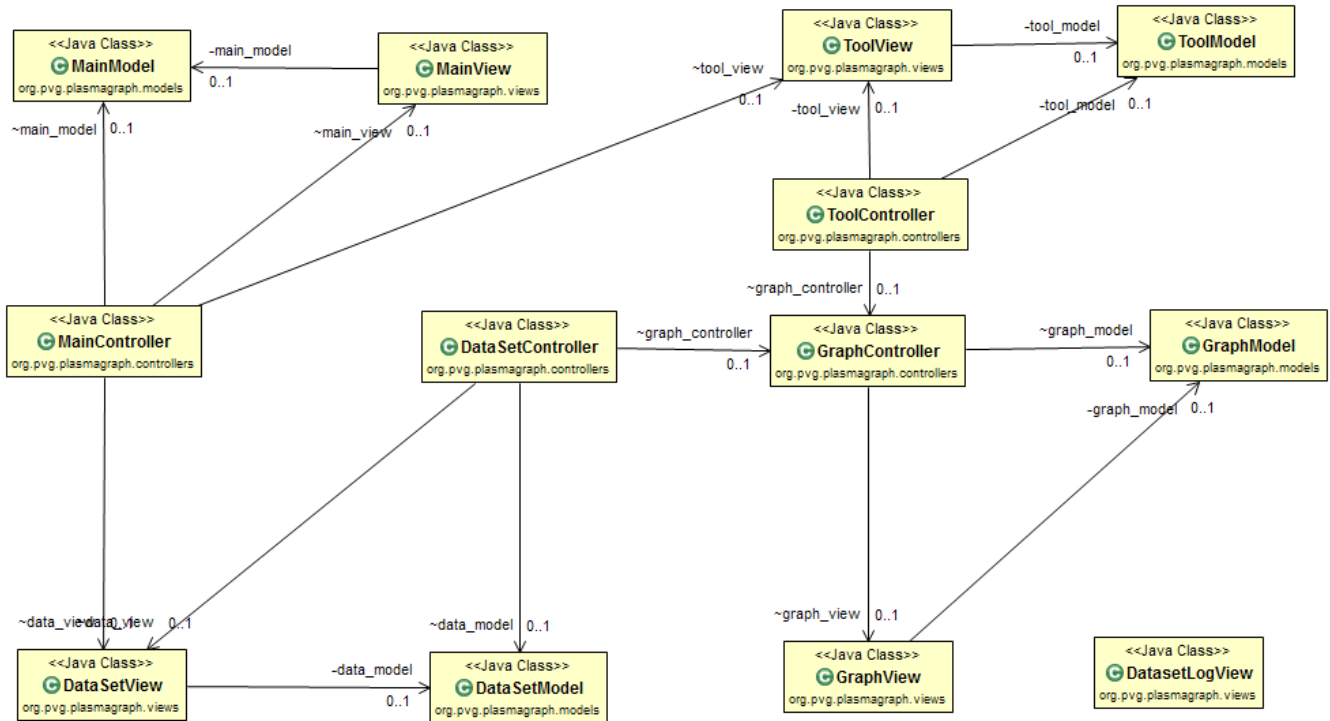
3.2.1.4. MVC – Graph MVC Class Relationships

Figure 5: Graph MVC Class Relationships



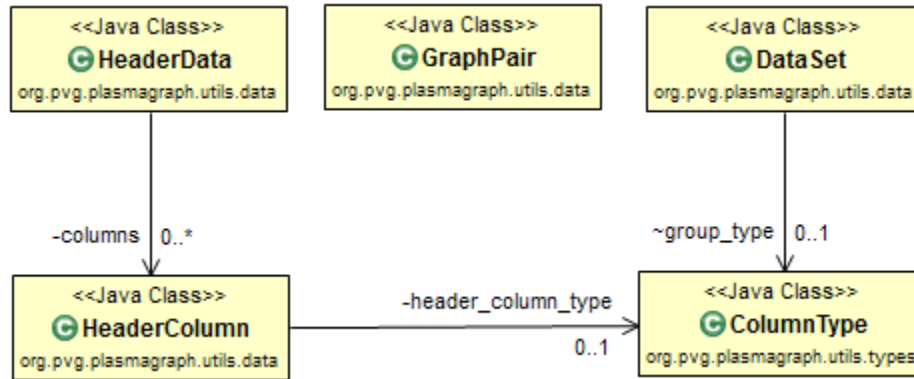
3.2.1.5. MVC – General MVC Class Relationships

Figure 6: General MVC Class Relationships



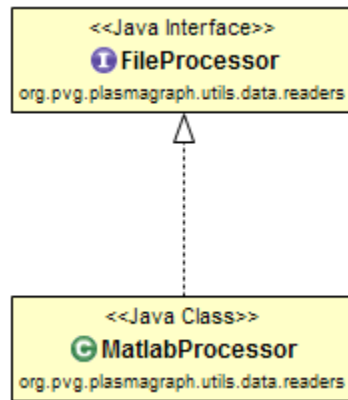
3.2.1.6. Data Class Relationships

Figure 7: Data Class Relationships



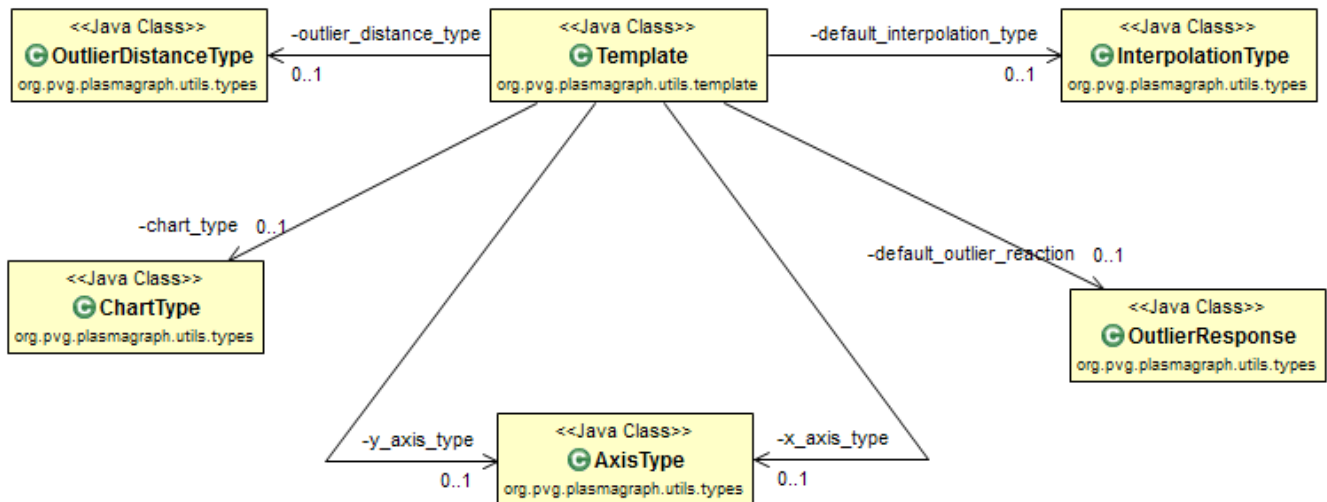
3.2.1.7. Data Reader Class Relationships

Figure 8: Data Reader Class Relationships



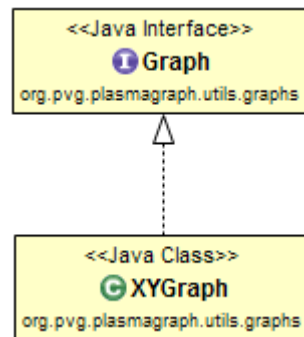
3.2.1.8. Template Class Relationships

Figure 9: Template Class Relationships



3.2.1.9. Graph Class Relationships

Figure 10: Graph Class Relationships



3.2.1.10. Outlier Searching Class Relationships











Figure 11: Outlier Searching Class Relationships



3.2.2. Package Contents

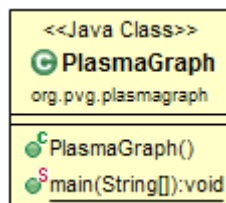
All diagrams in this section will use the following icons to refer to various components of the code.

Figure 12: Class Contents Diagram - Symbol Terminology Legend

Icon	Meaning
	Class
	Interface
	Attribute, private
	Attribute, public
	Method, constructor
	Method, final
	Method, main
	Method, override
	Method, private
	Method, public

3.2.2.1. Package "org.pvg.plasmagraph"

Figure 13: PlasmaGraph Class Contents Diagram



3.2.2.2. Package "org.pvg.plasmagraph.controllers"

Figure 14: MainController Class Contents Diagram

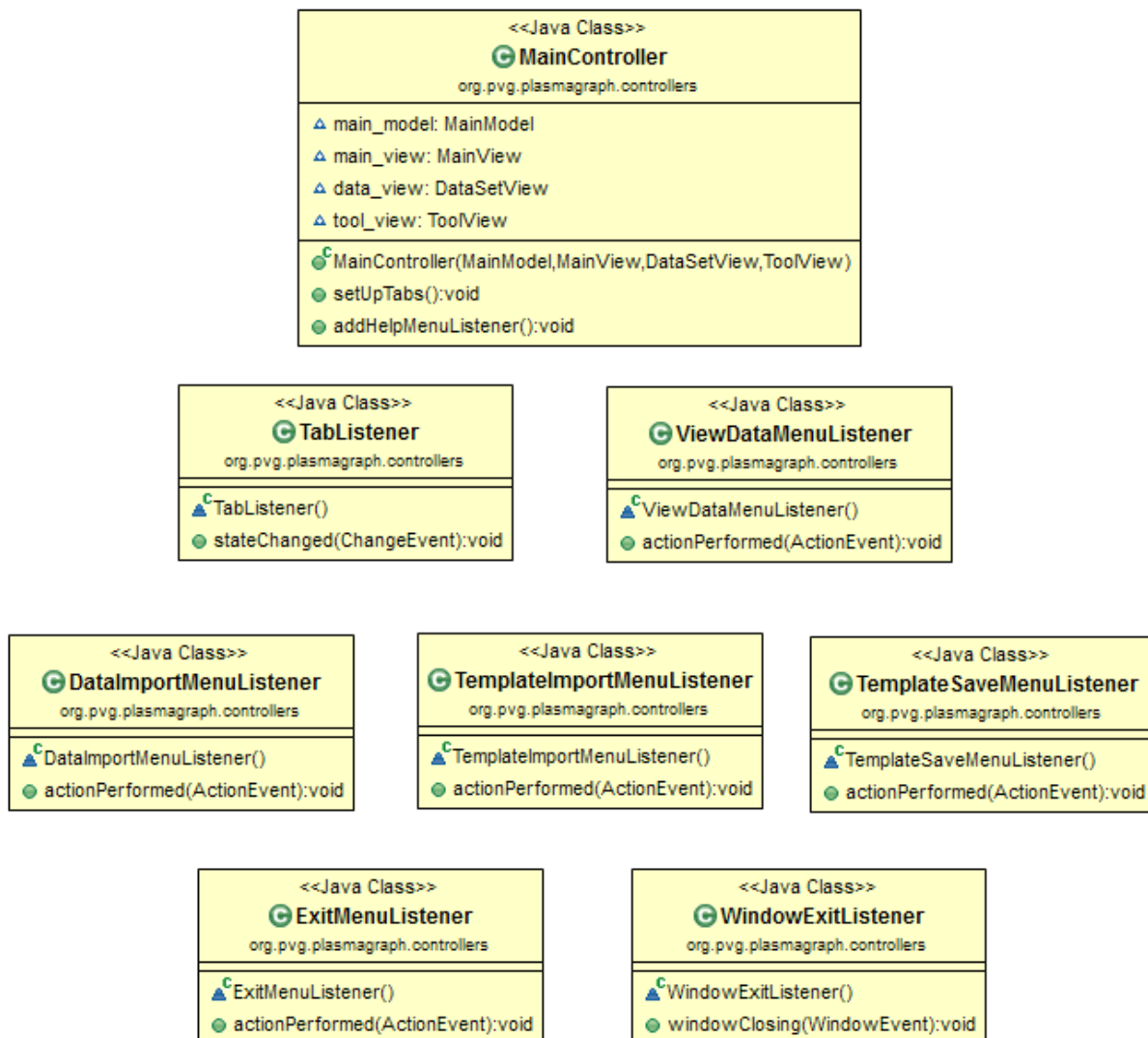


Figure 15: DataSetController Class Contents Diagram

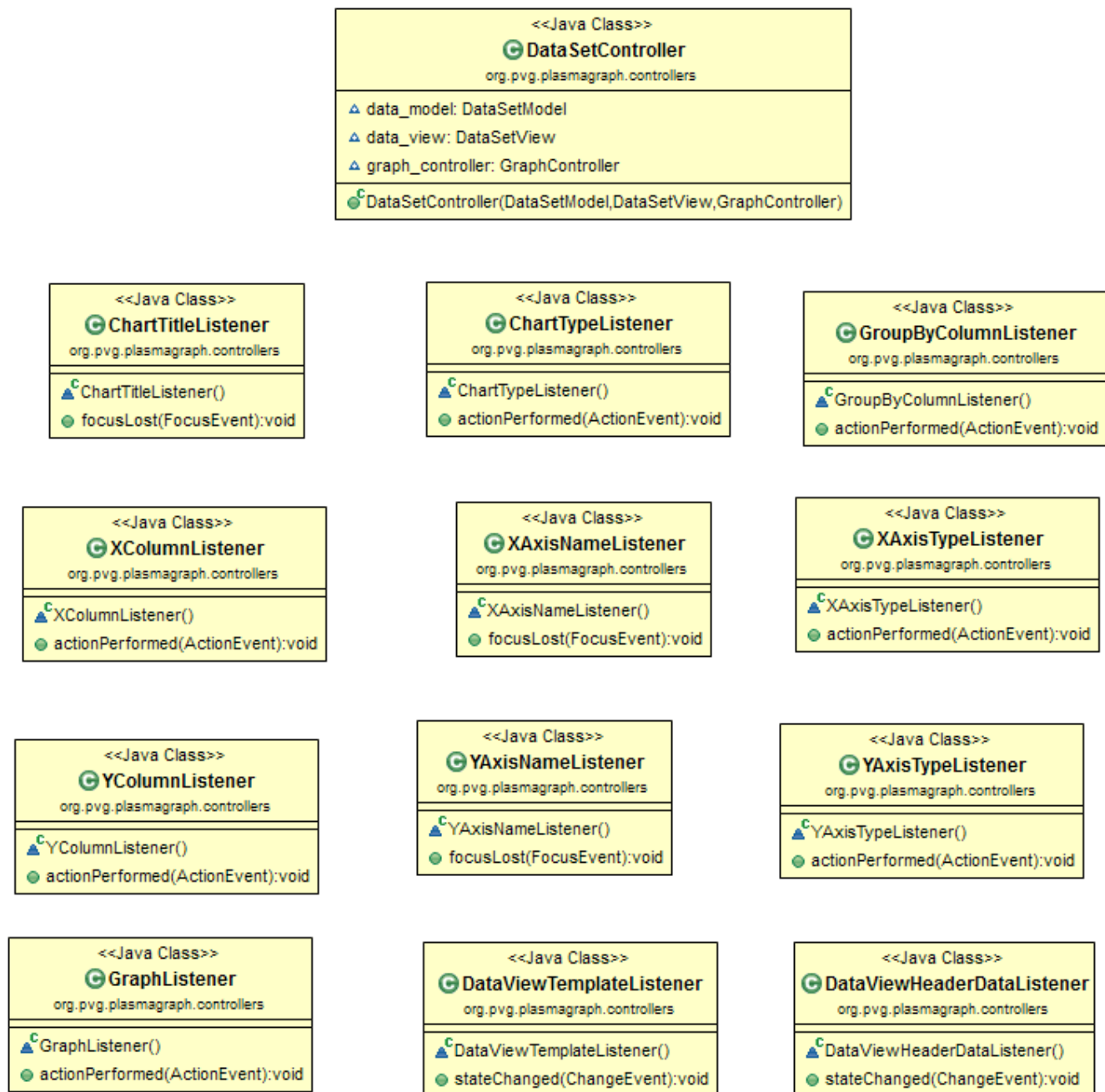


Figure 16: ToolController Class Contents Diagram

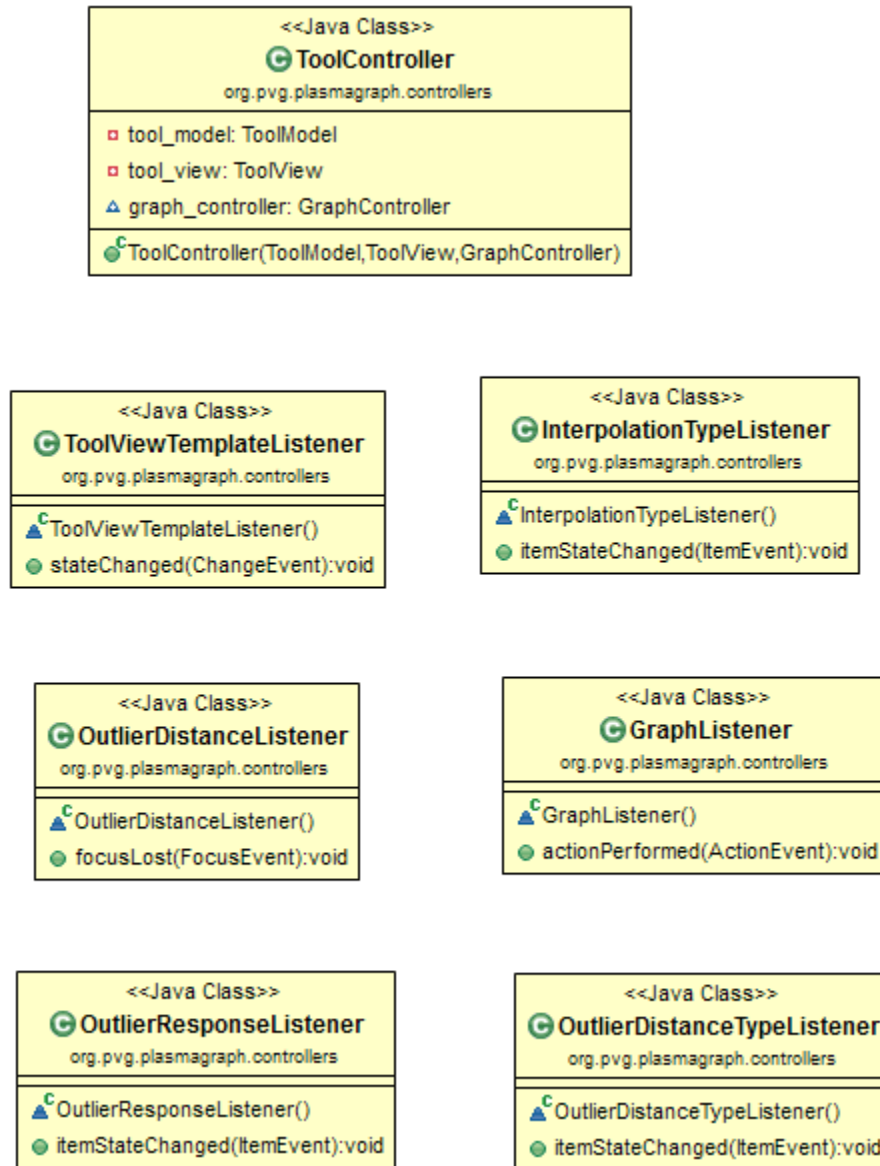
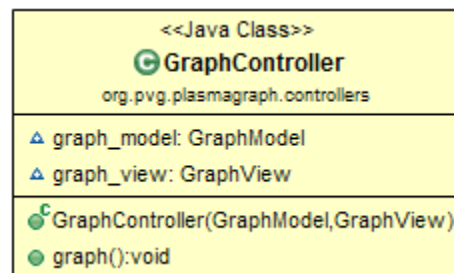


Figure 17: GraphController Class Contents Diagram



3.2.2.3. Package "org.pvg.plasmagraph.models"

Figure 18: MainModel Class Contents Diagram

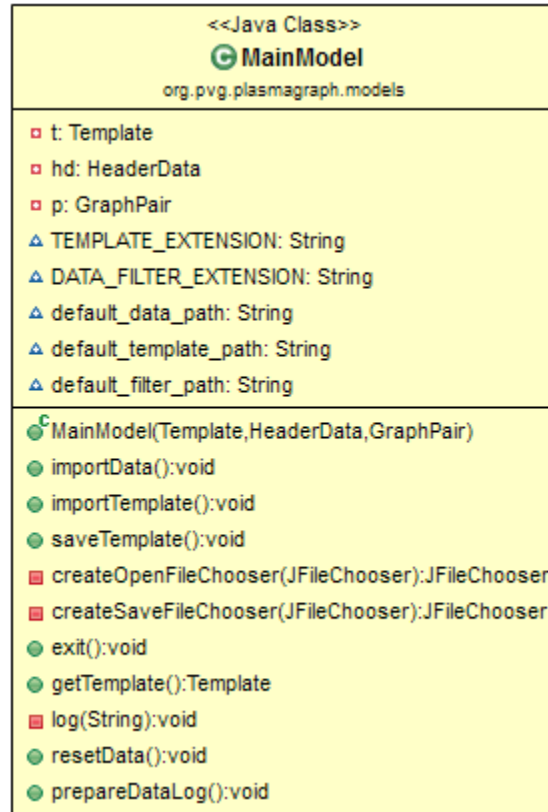


Figure 19: DataSetModel Class Contents Diagram

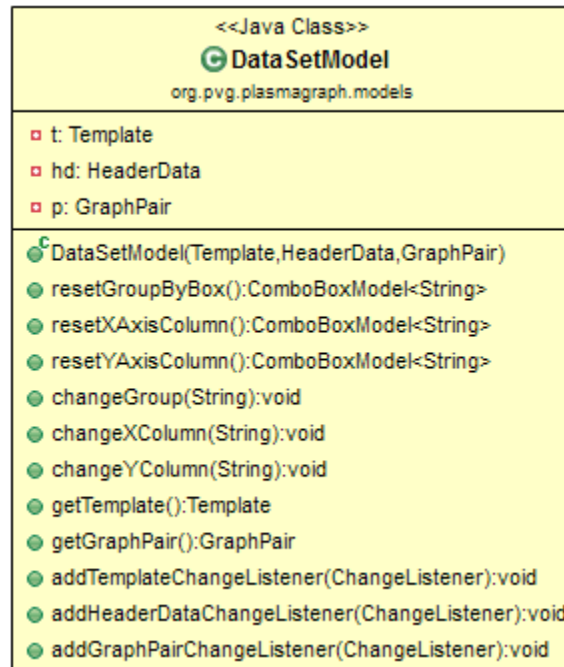


Figure 20: ToolModel Class Contents Diagram

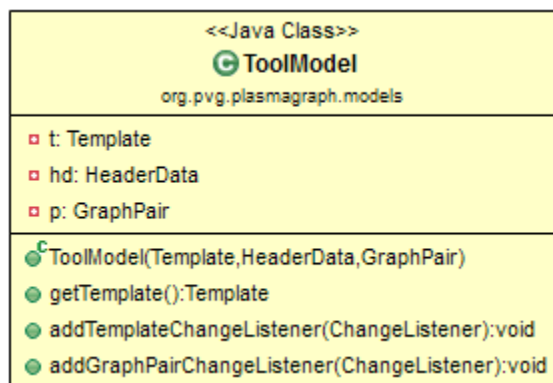
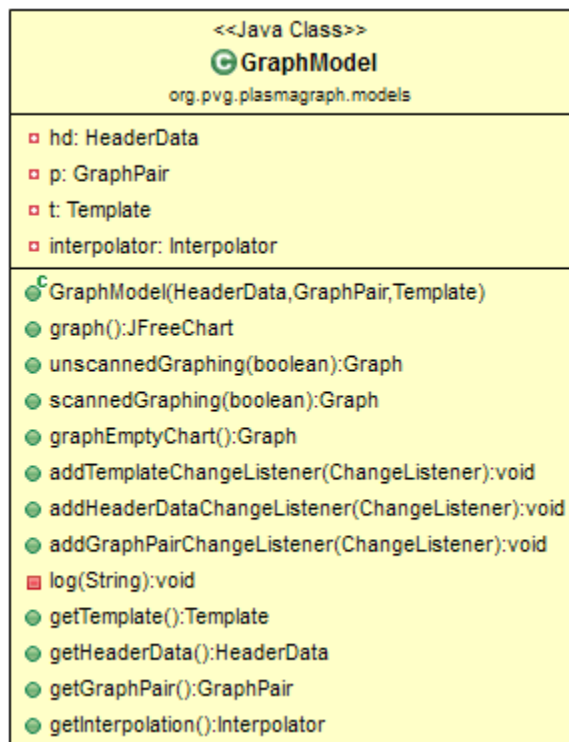
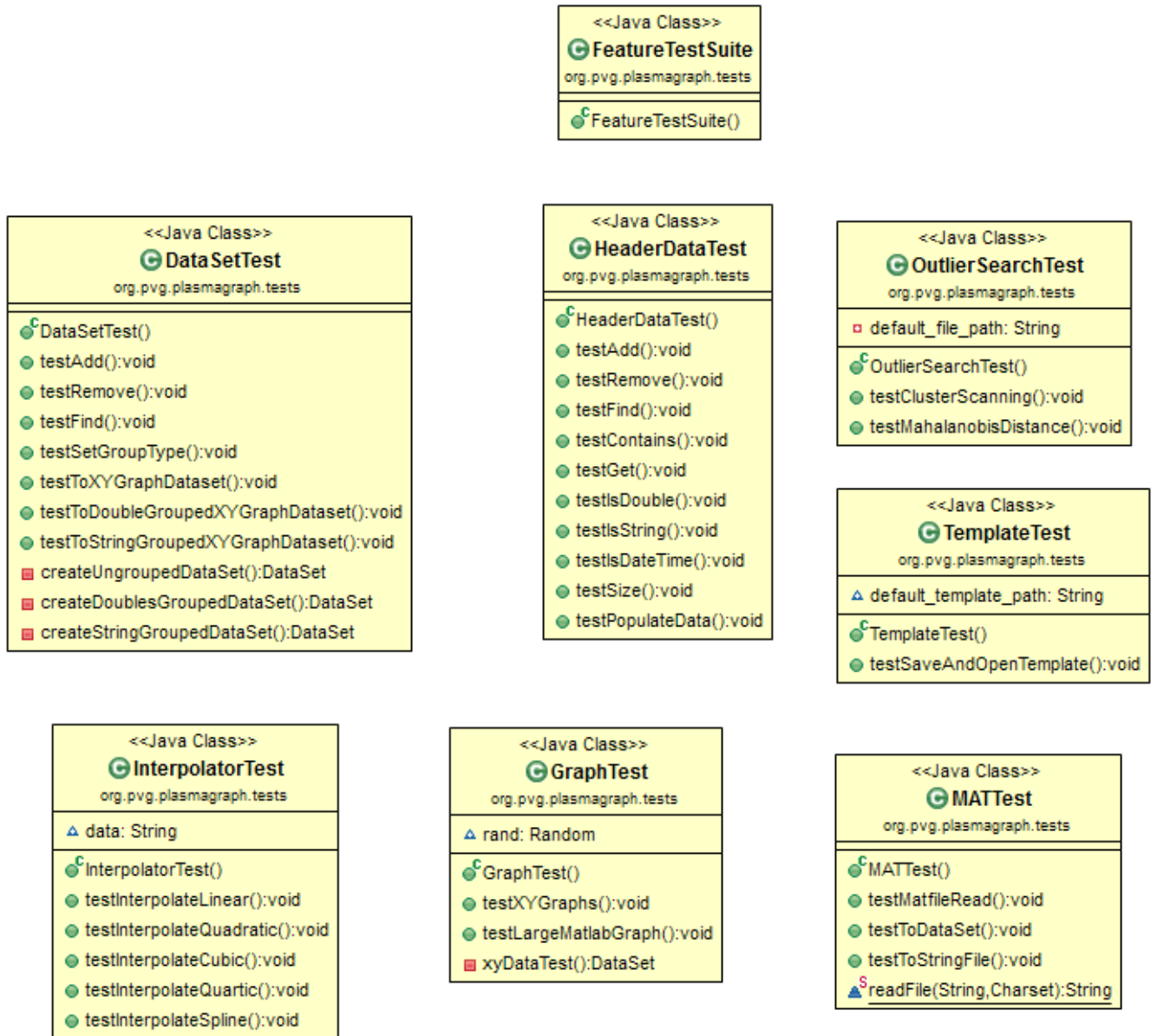


Figure 21: GraphModel Class Contents Diagram



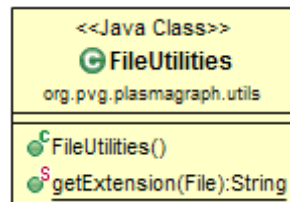
3.2.2.4. Package "org.pvg.plasmagraph.tests"

Figure 22: Automated Test Class Contents Diagram



3.2.2.5. Package "org.pvg.plasmagraph.utils"

Figure 23: FileUtilities Class Contents Diagram



3.2.2.6. Package "org.pvg.plasmagraph.utils.data"

Figure 24: HeaderData Class Contents Diagram

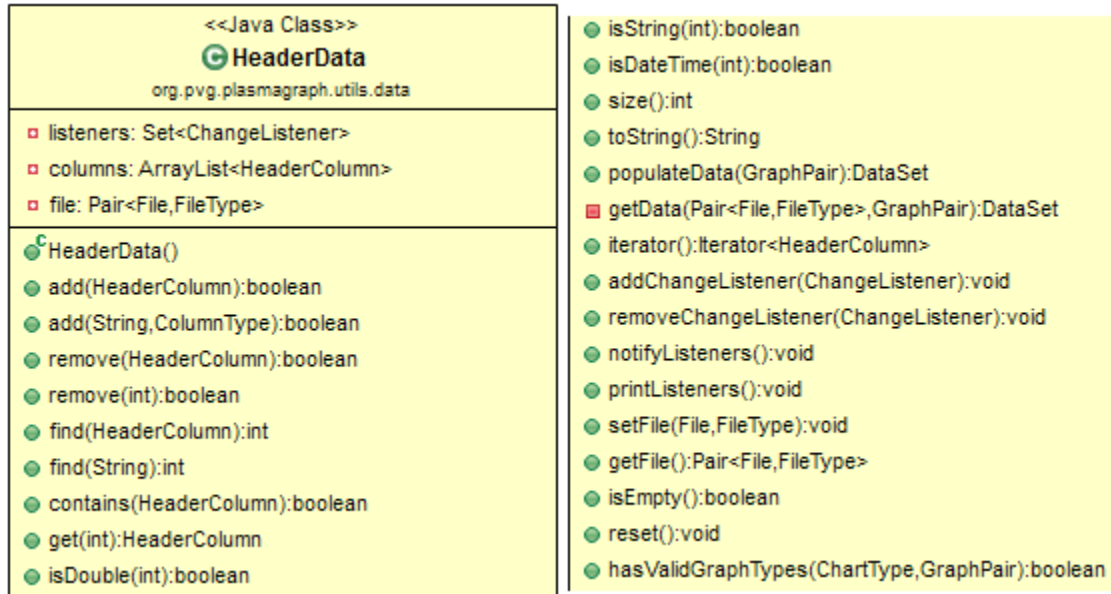


Figure 25: HeaderColumn Class Contents Diagram

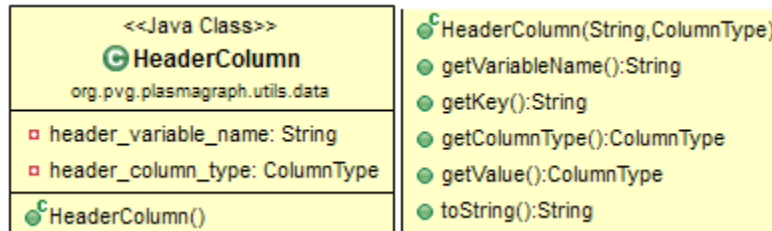


Figure 26: GraphPair Class Contents Diagram

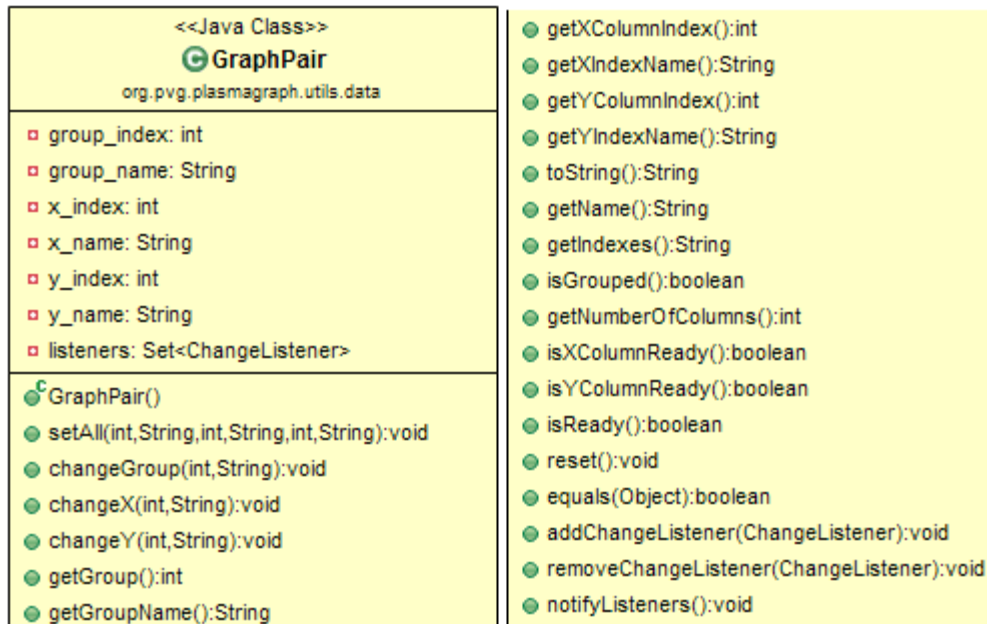
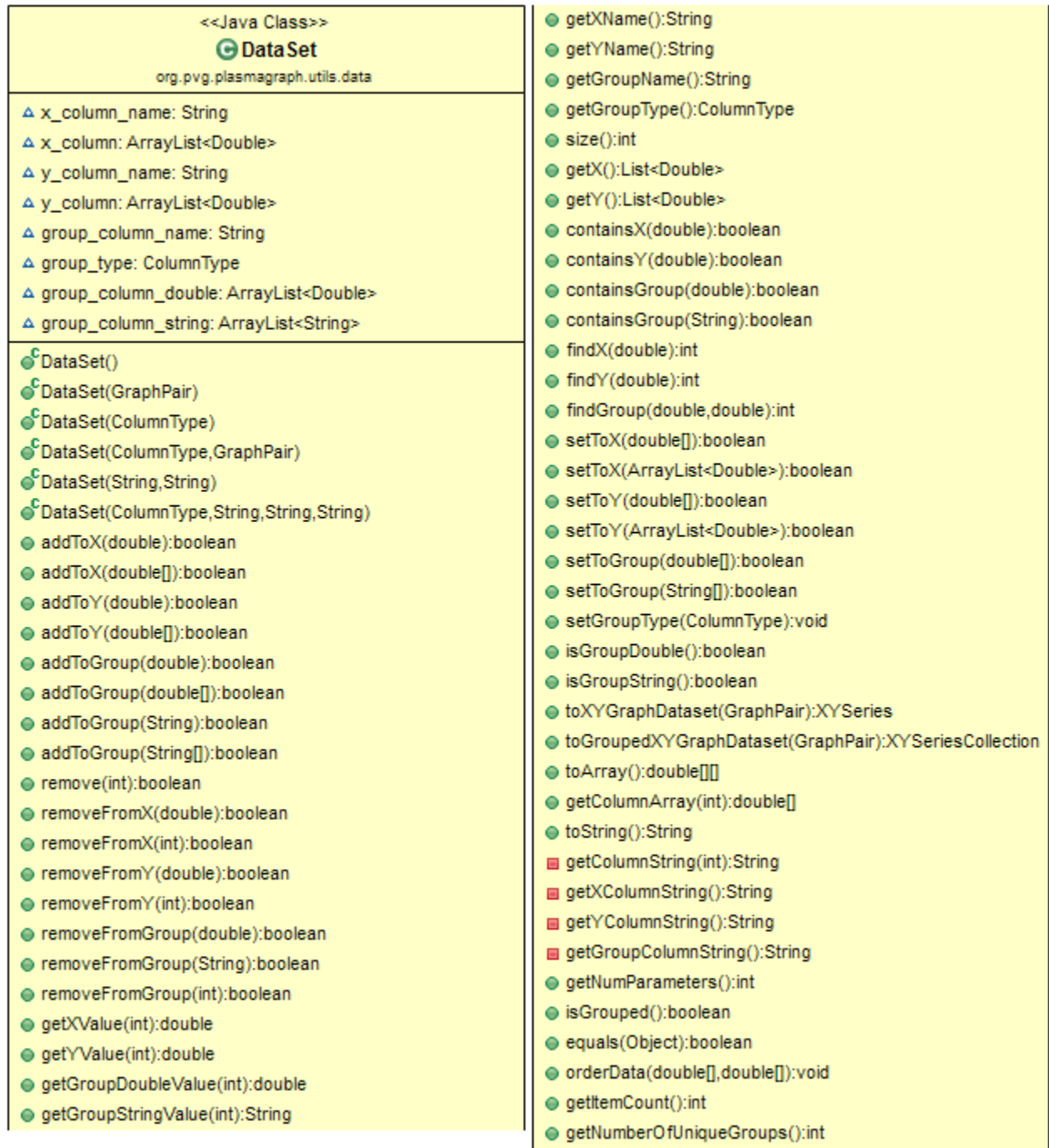


Figure 27: DataSet Class Contents Diagram



3.2.2.7. Package "org.pvg.plasmagraph.utils.data.readers"

Figure 28: FileProcessor Interface Contents Diagram

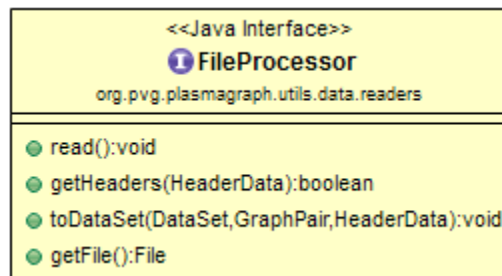
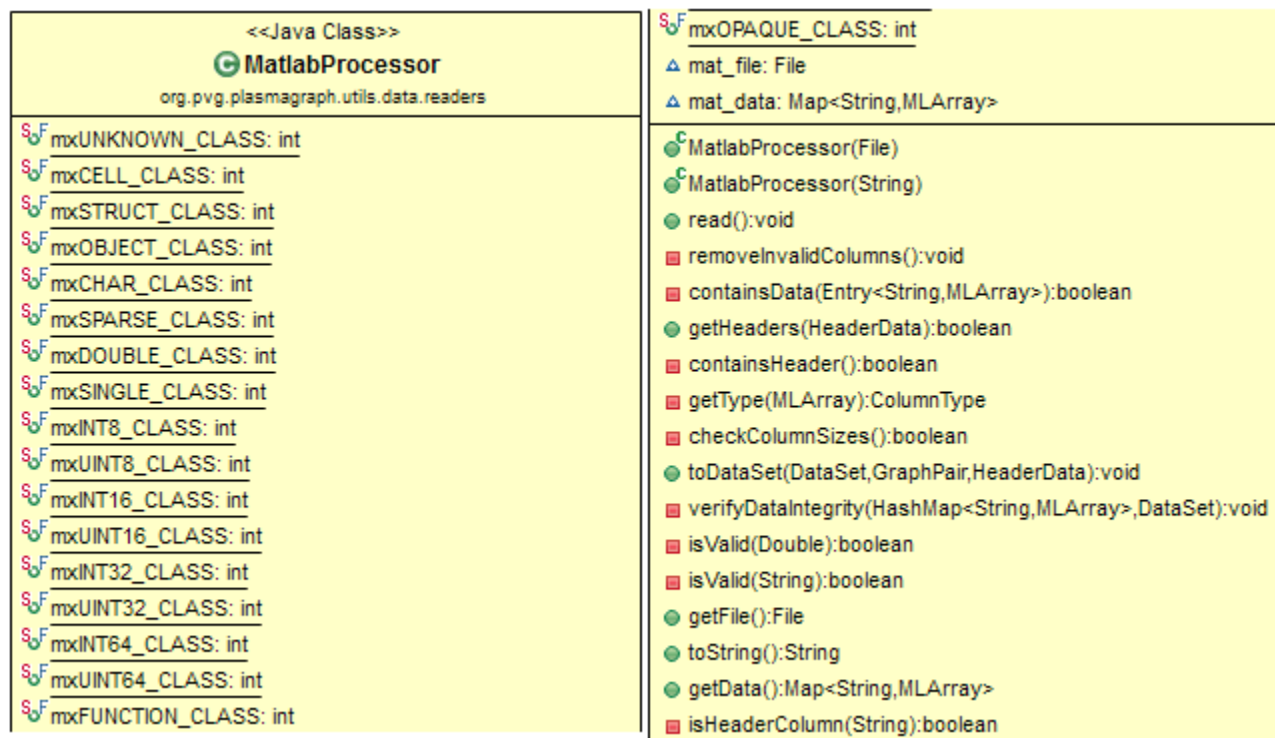


Figure 29: MatlabProcessor Class Contents Diagram



3.2.2.8. Package "org.pvg.plasmagraph.utils.graphs"

Figure 30: Graph Interface Contents Diagram

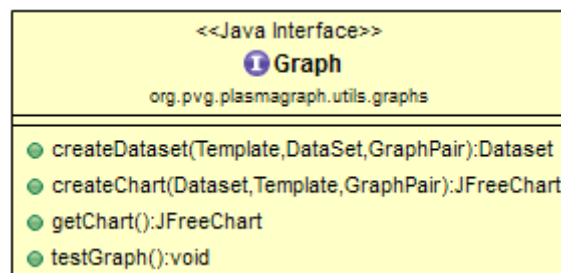
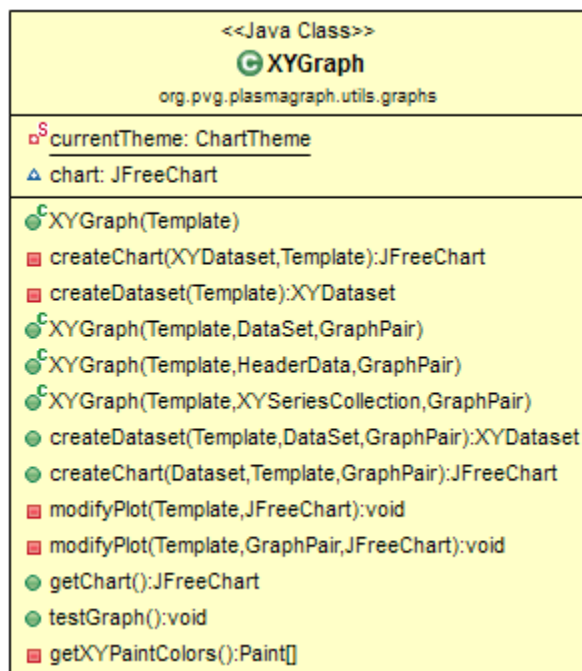
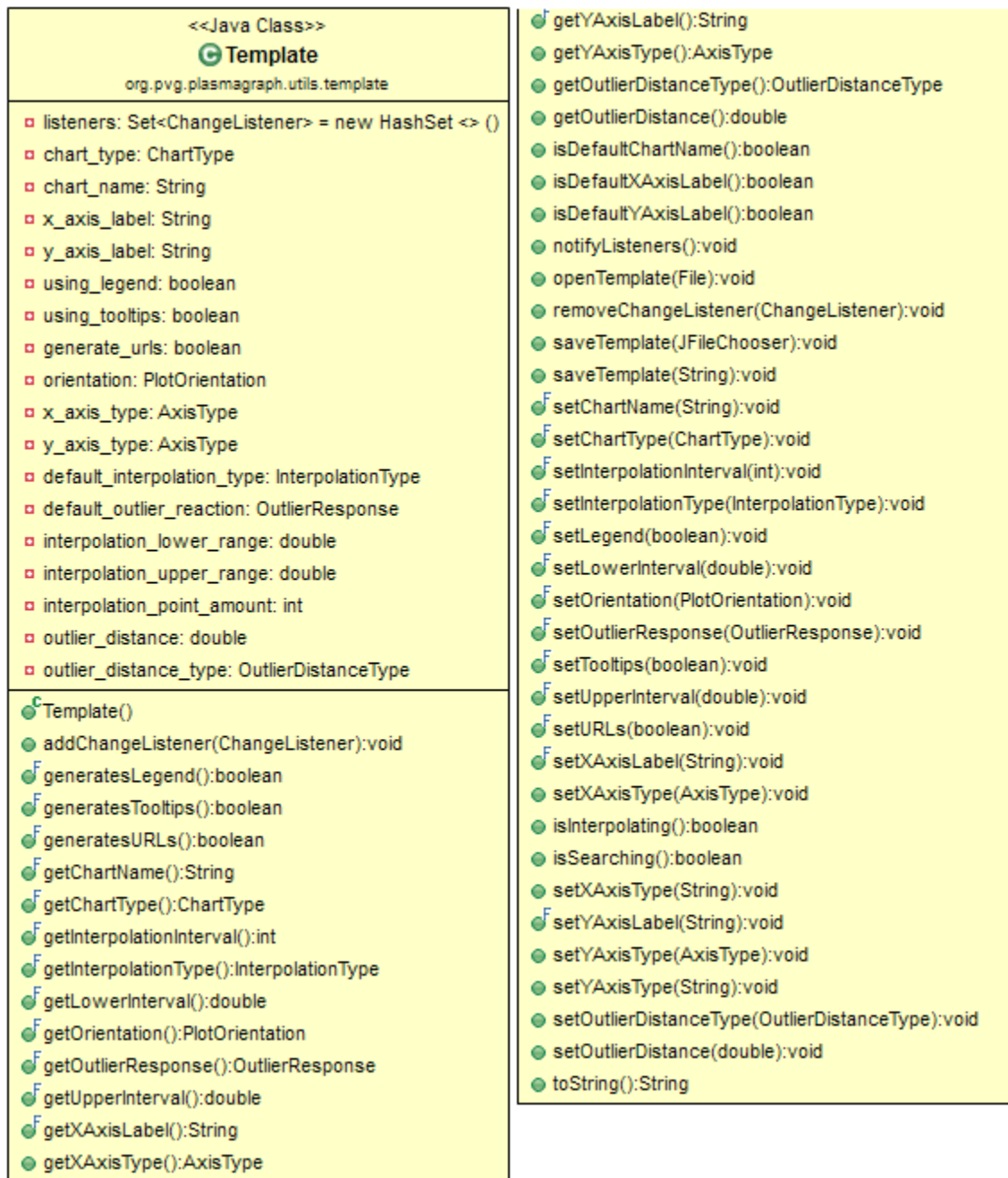


Figure 31: XYGraph Class Contents Diagram



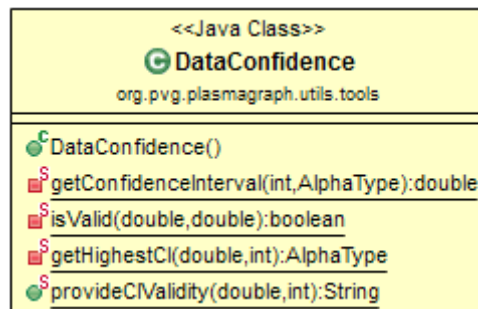
3.2.2.9. Package "org.pvg.plasmagraph.utils.template"

Figure 32: Template Class Contents Diagram



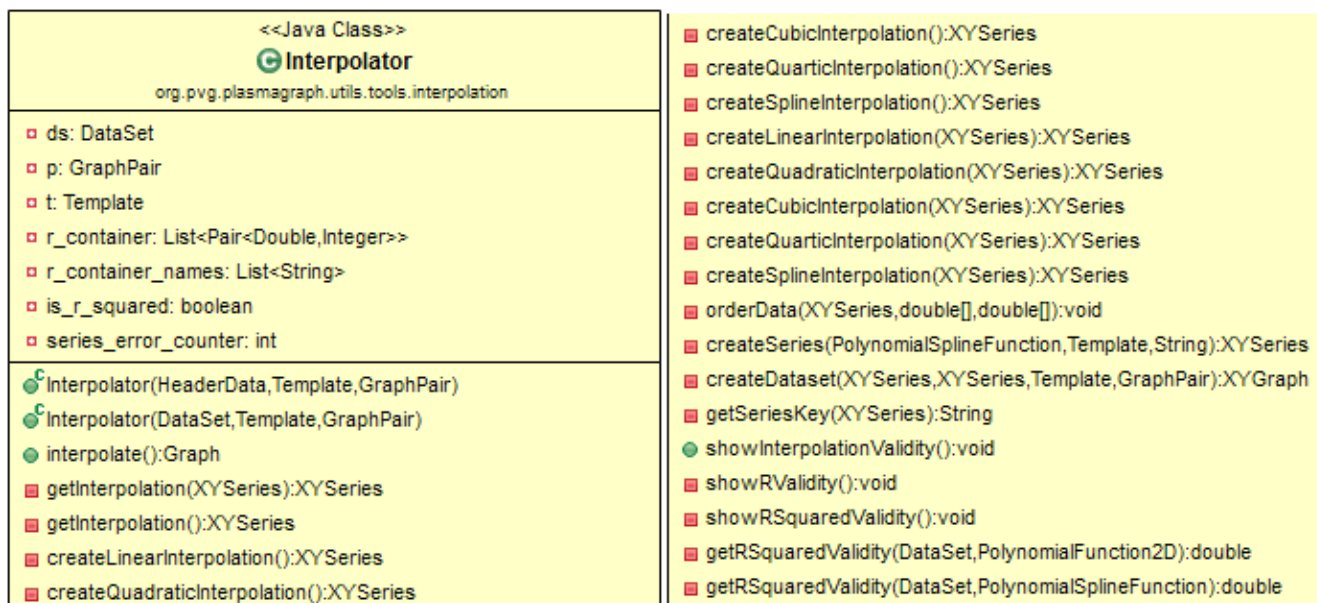
3.2.2.10. Package "org.pvg.plasmagraph.utils.tools"

Figure 33: DataConfidence Class Contents Diagram



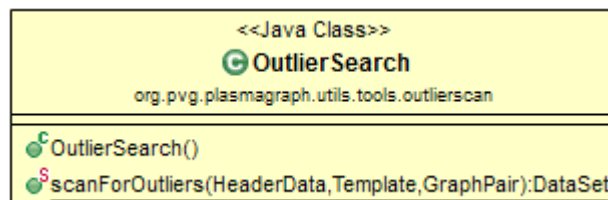
3.2.2.11. Package "org.pvg.plasmagraph.utils.tools.interpolation"

Figure 34: Interpolator Class Contents Diagram



3.2.2.12. Package "org.pvg.plasmagraph.utils.tools.outlierscan"

Figure 35: OutlierSearch Class Contents Diagram



3.2.2.13. Package "org.pvg.plasmagraph.utils.tools.outlierscan.distances"

Figure 36: OutlierDistance Interface Contents Diagram

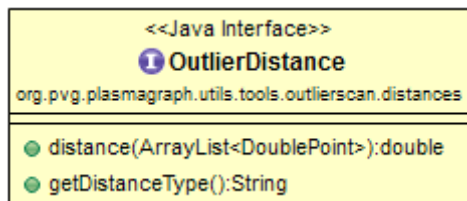


Figure 37: CartesianDistance Class Contents Diagram

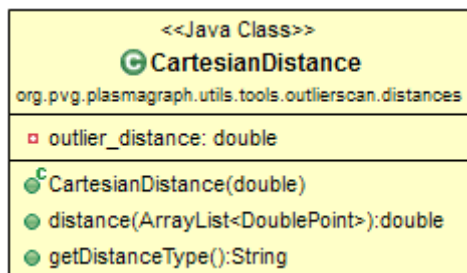
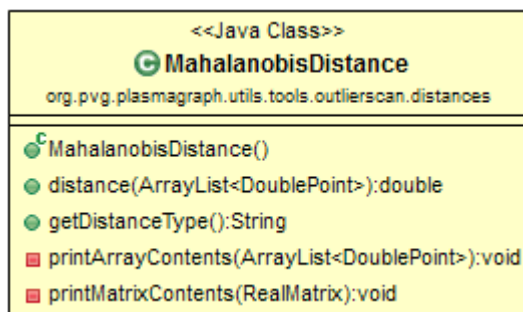


Figure 38: MahalanobisDistance Class Contents Diagram



3.2.2.14. Package "org.pvg.plasmagraph.utils.tools.outlierscan.methods"

Figure 39: ScanMethod Interface Contents Diagram

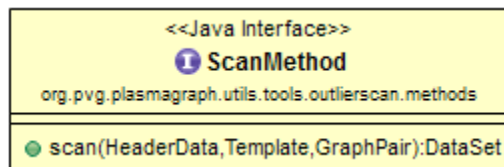
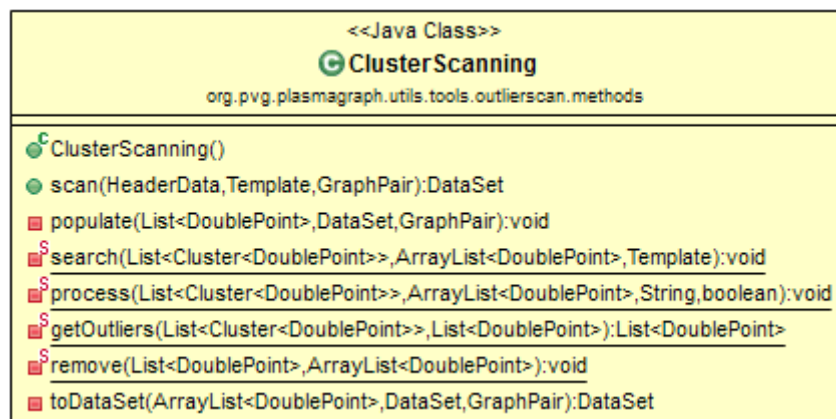
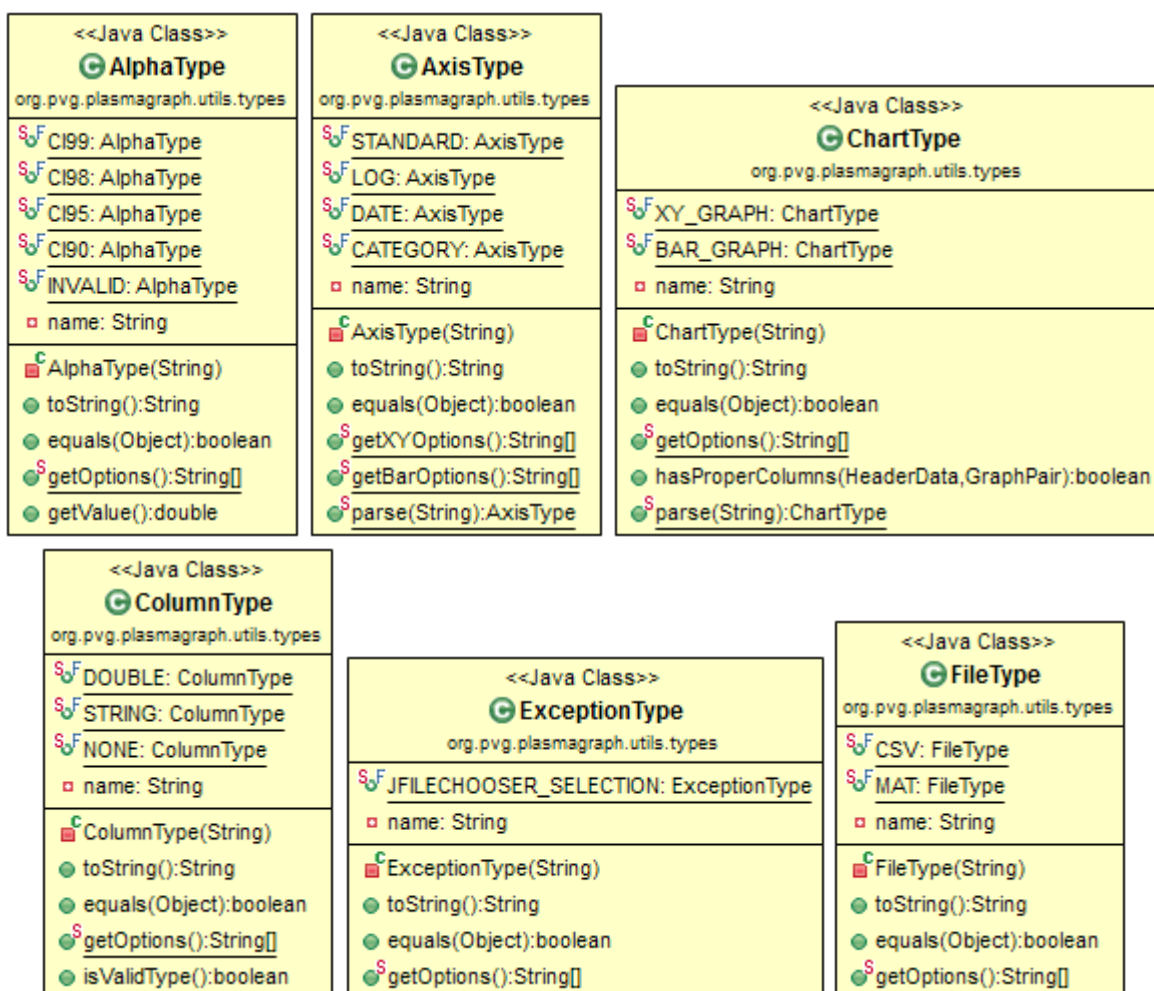


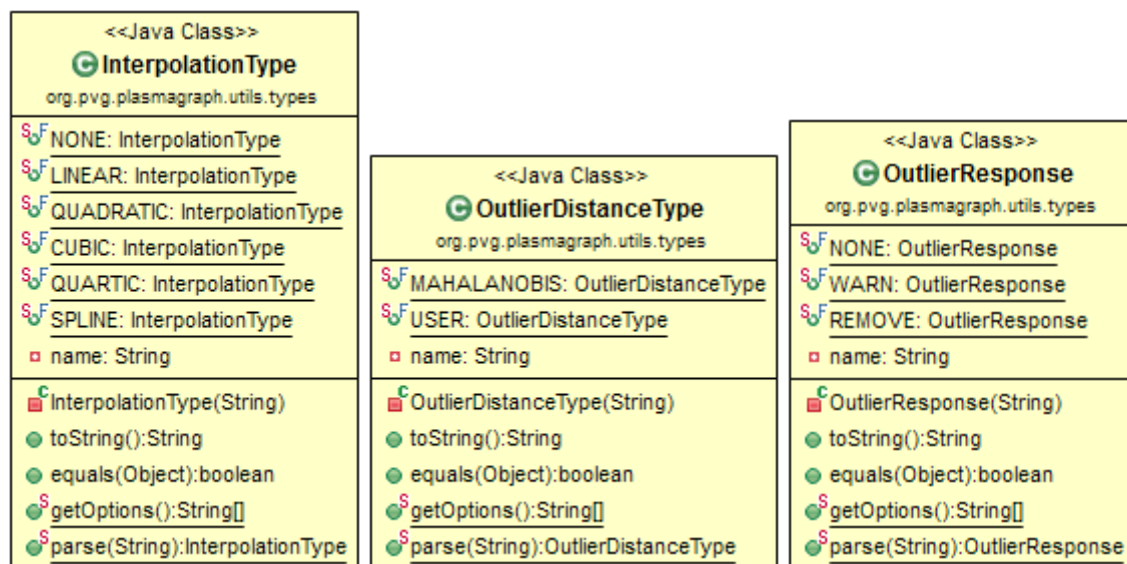
Figure 40: ClusterScanning Class Contents Diagram



3.2.2.15. Package "org.pvg.plasmagraph.utils.tools.types"

Figure 41: Data Type Class Contents Diagrams





3.2.2.16. Package "org.pvg.plasmagraph.views"

Figure 42: MainView Class Contents Diagram

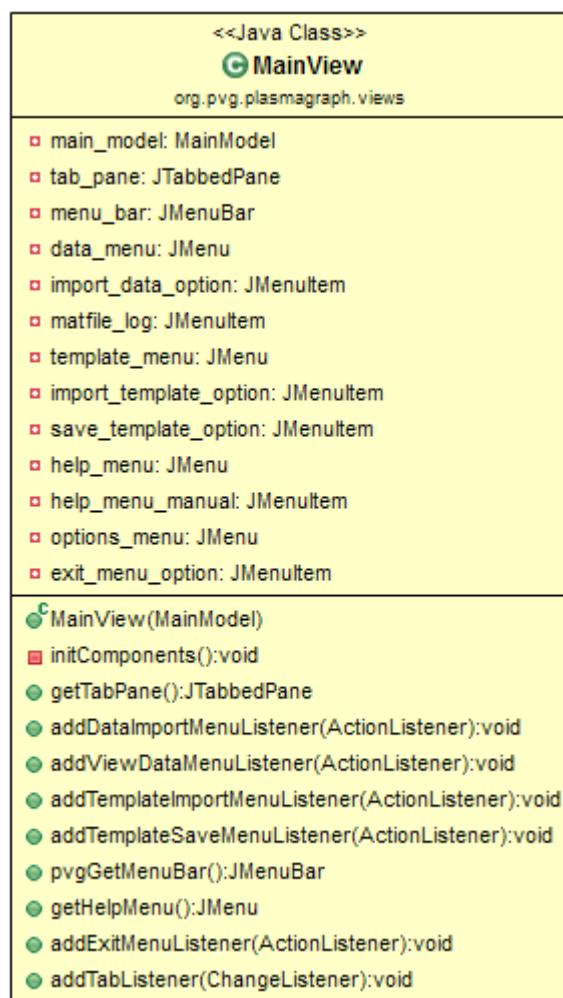


Figure 43: DataSetView Class Contents Diagram

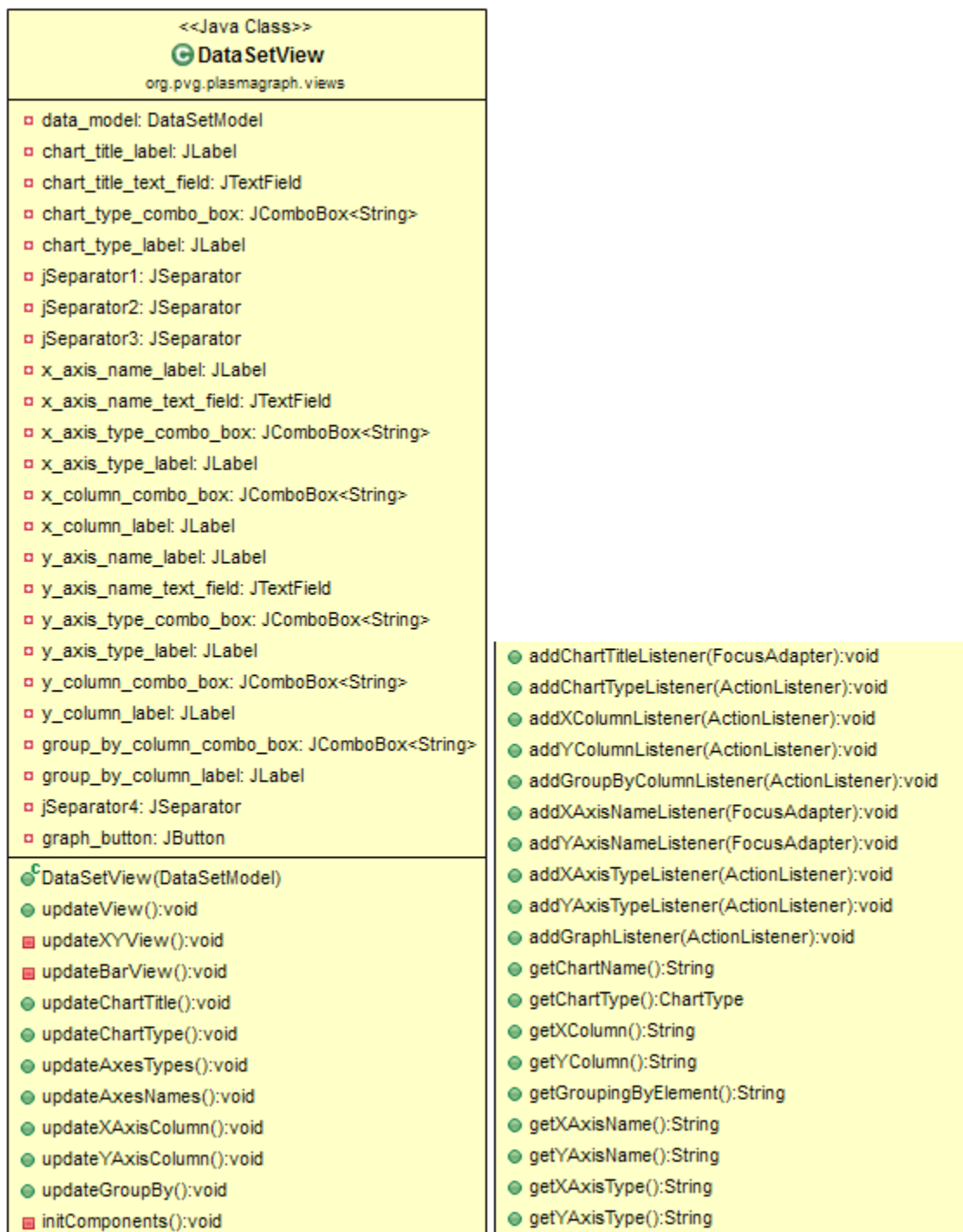


Figure 44: ToolView Class Contents Diagram

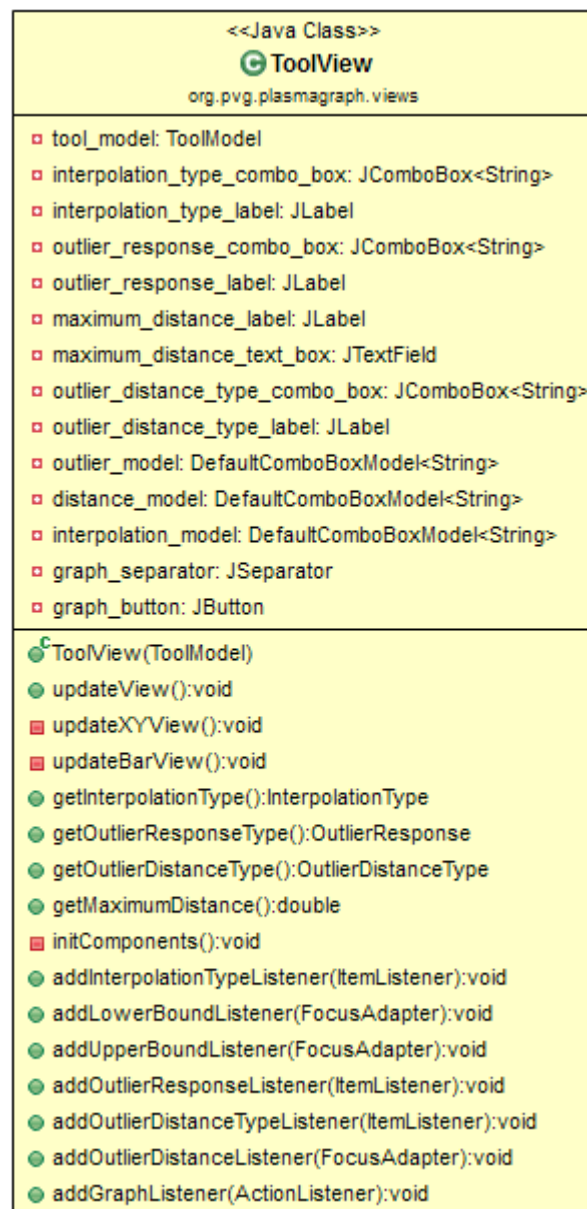


Figure 45: GraphView Class Contents Diagram

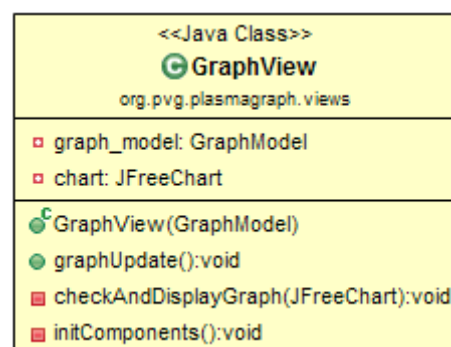
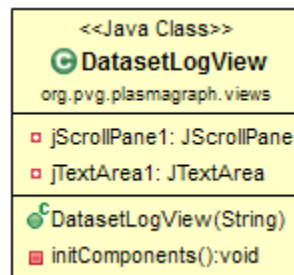
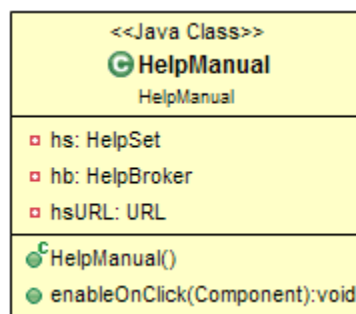


Figure 46: DatasetLogView Class Contents Diagram

3.2.2.17. Package "HelpManual"

Figure 47: Help Manual Class Contents Diagram

3.2.3. Requirement Fulfillment

3.2.3.1. Functional Requirements

Figure 48: "Import Data" Sequence Diagram

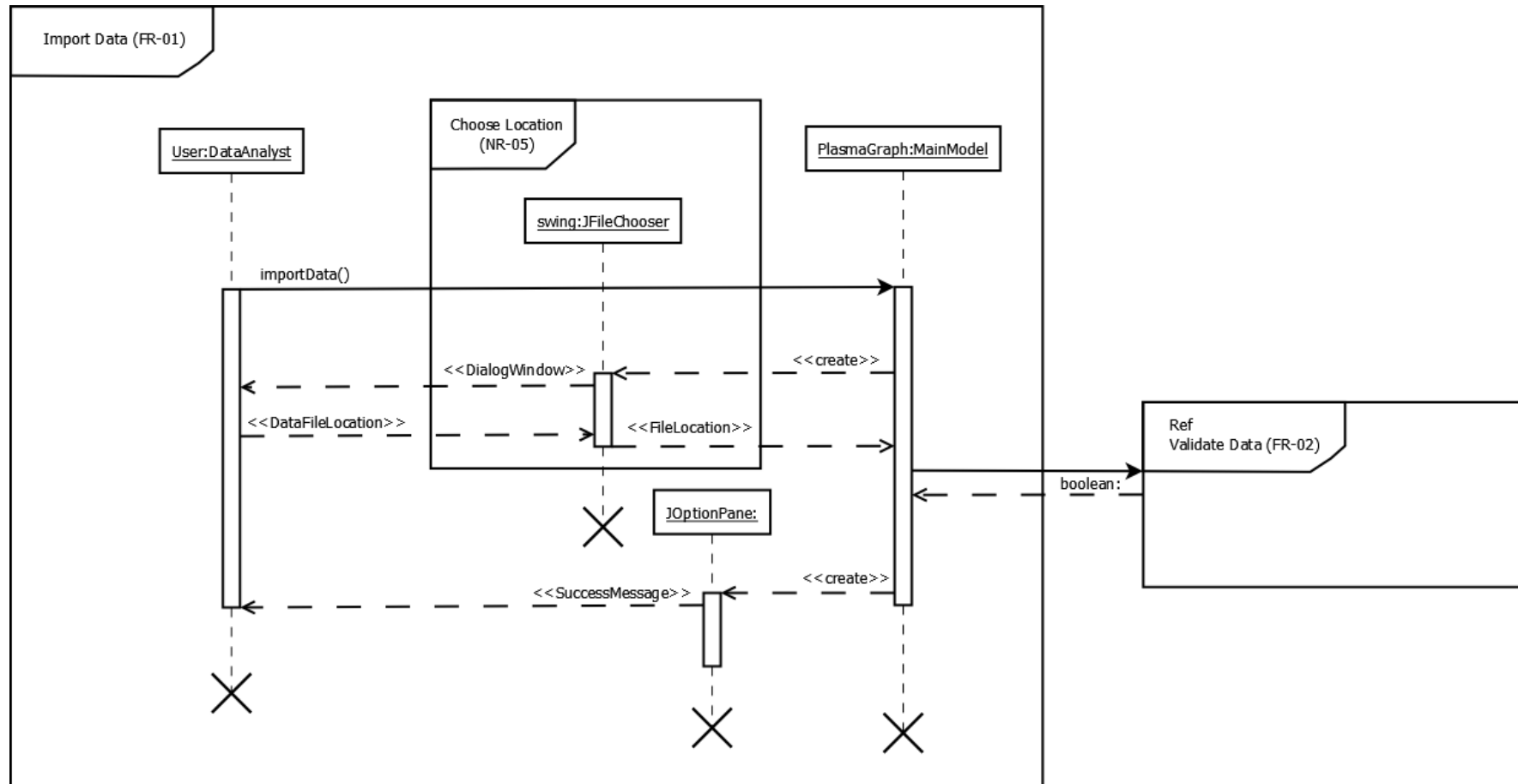


Figure 49: “Validate Data” Sequence Diagram

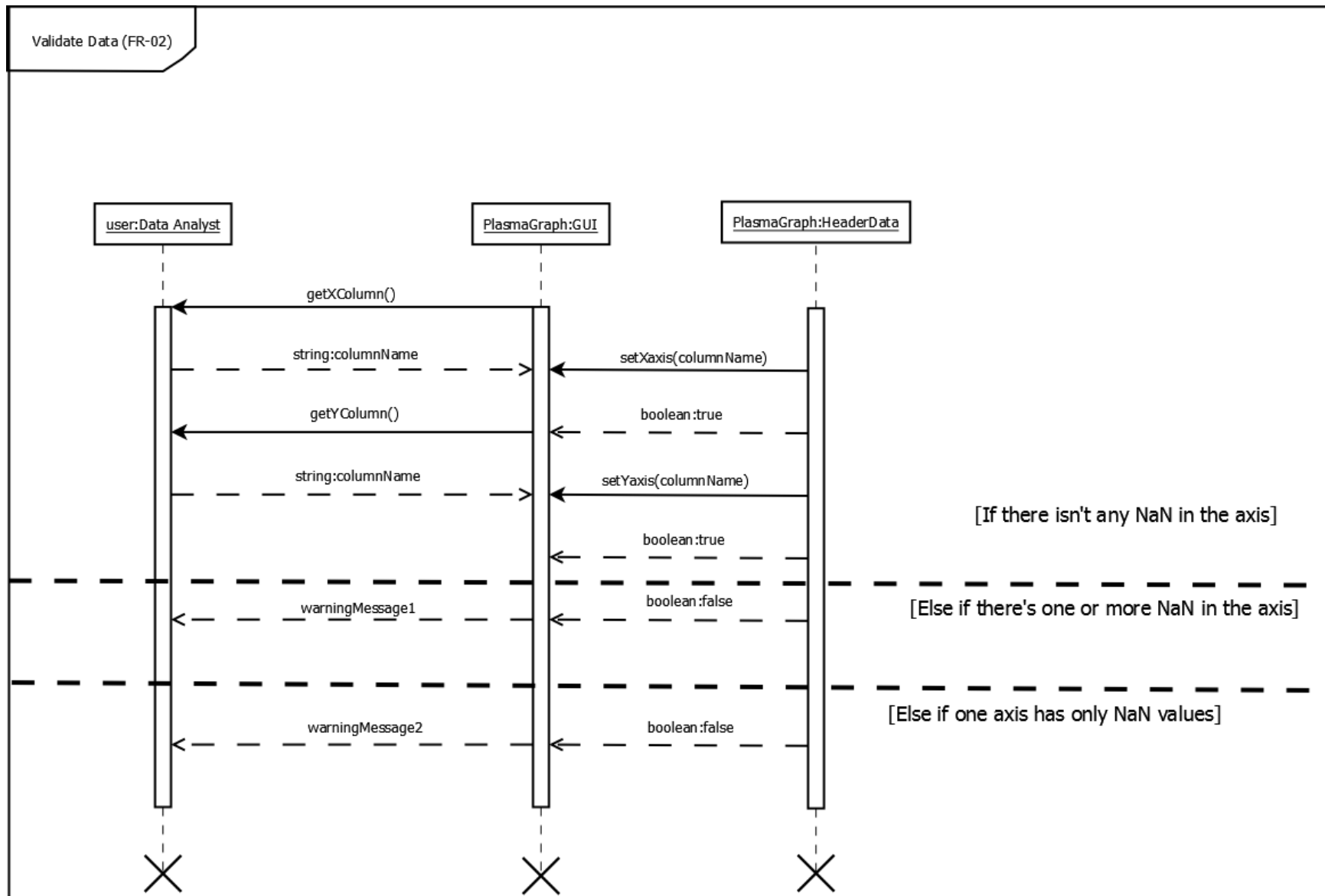


Figure 50: “Choose Graph Options” Sequence Diagram

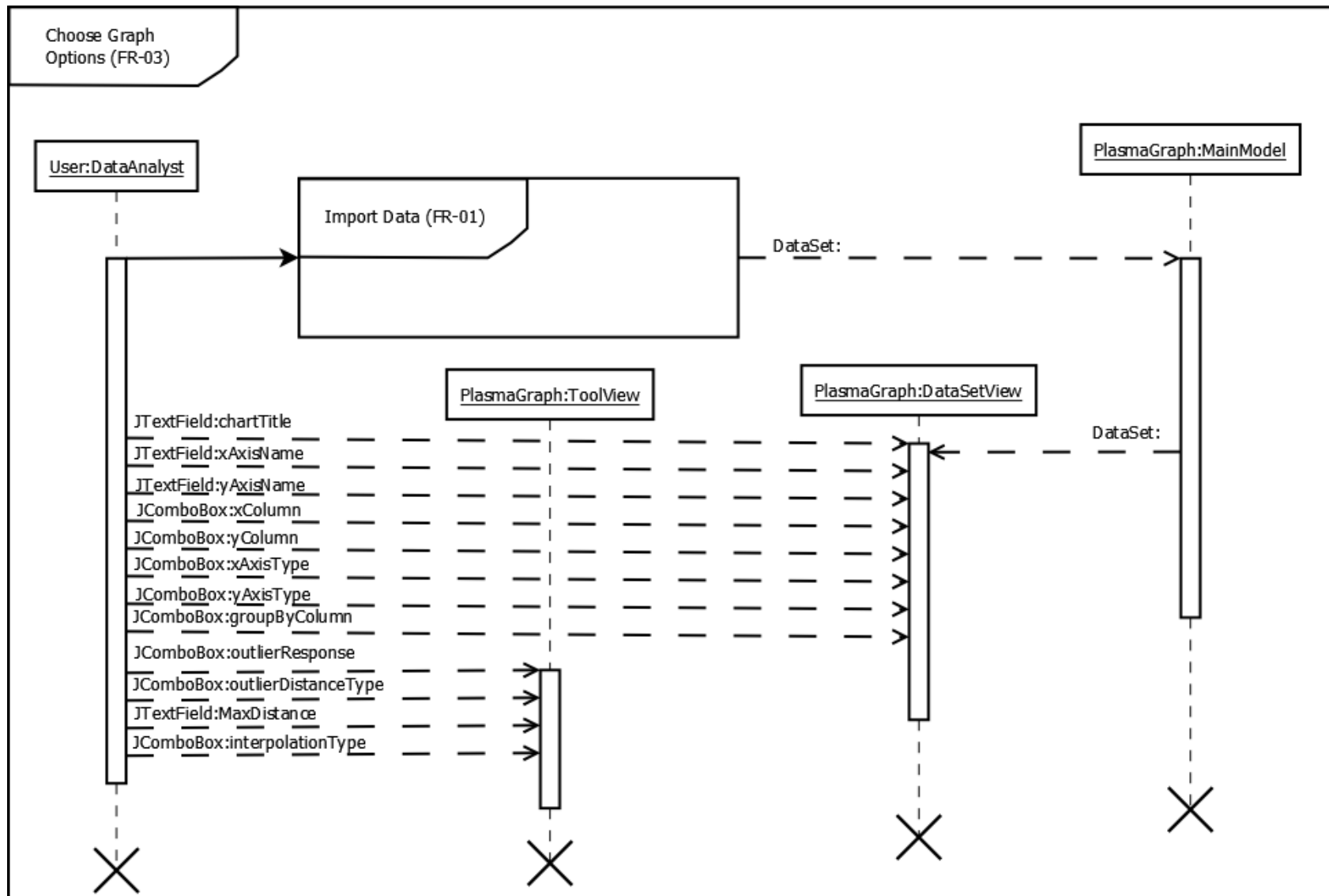


Figure 51: "Create Graph" Sequence Diagram

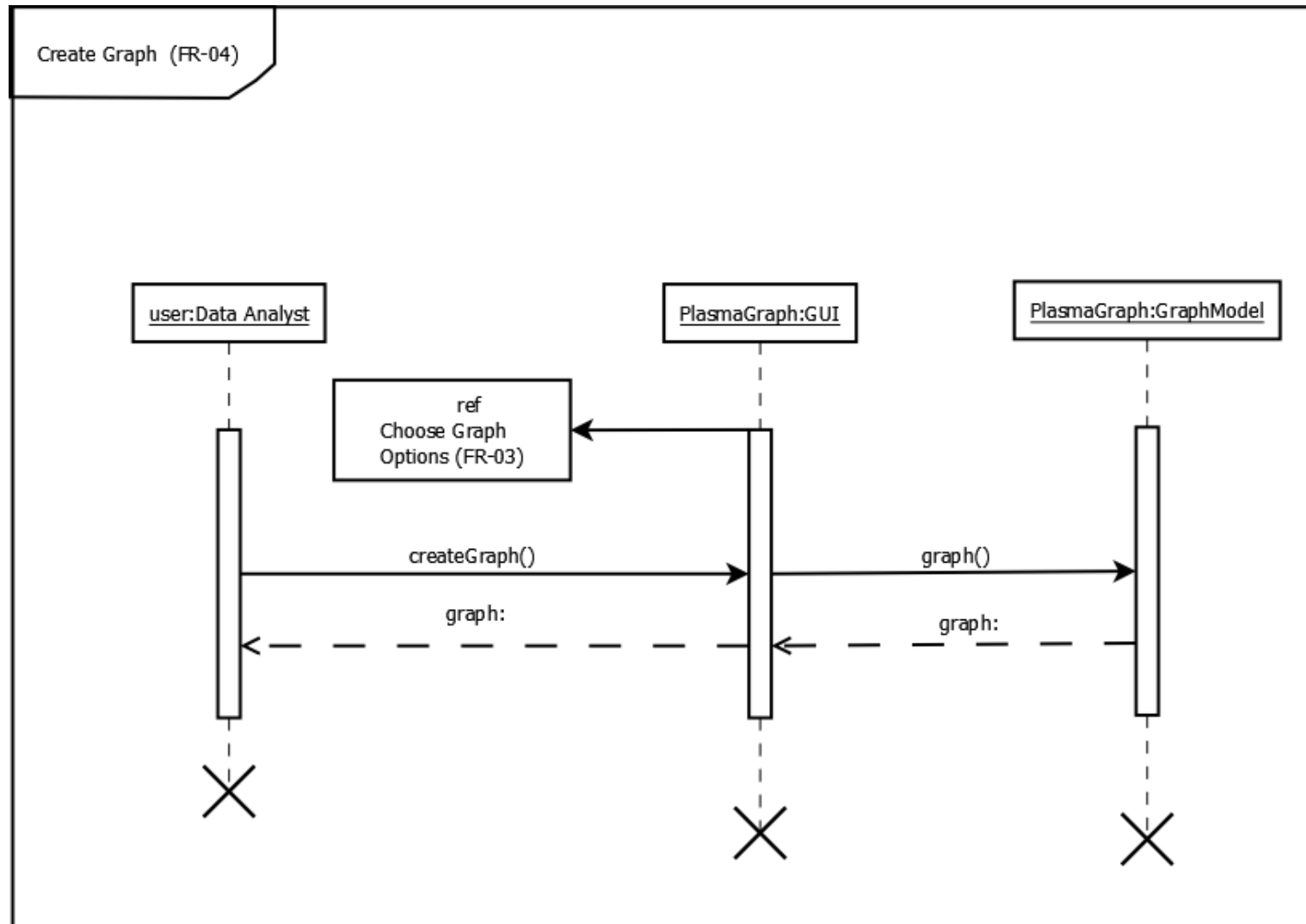
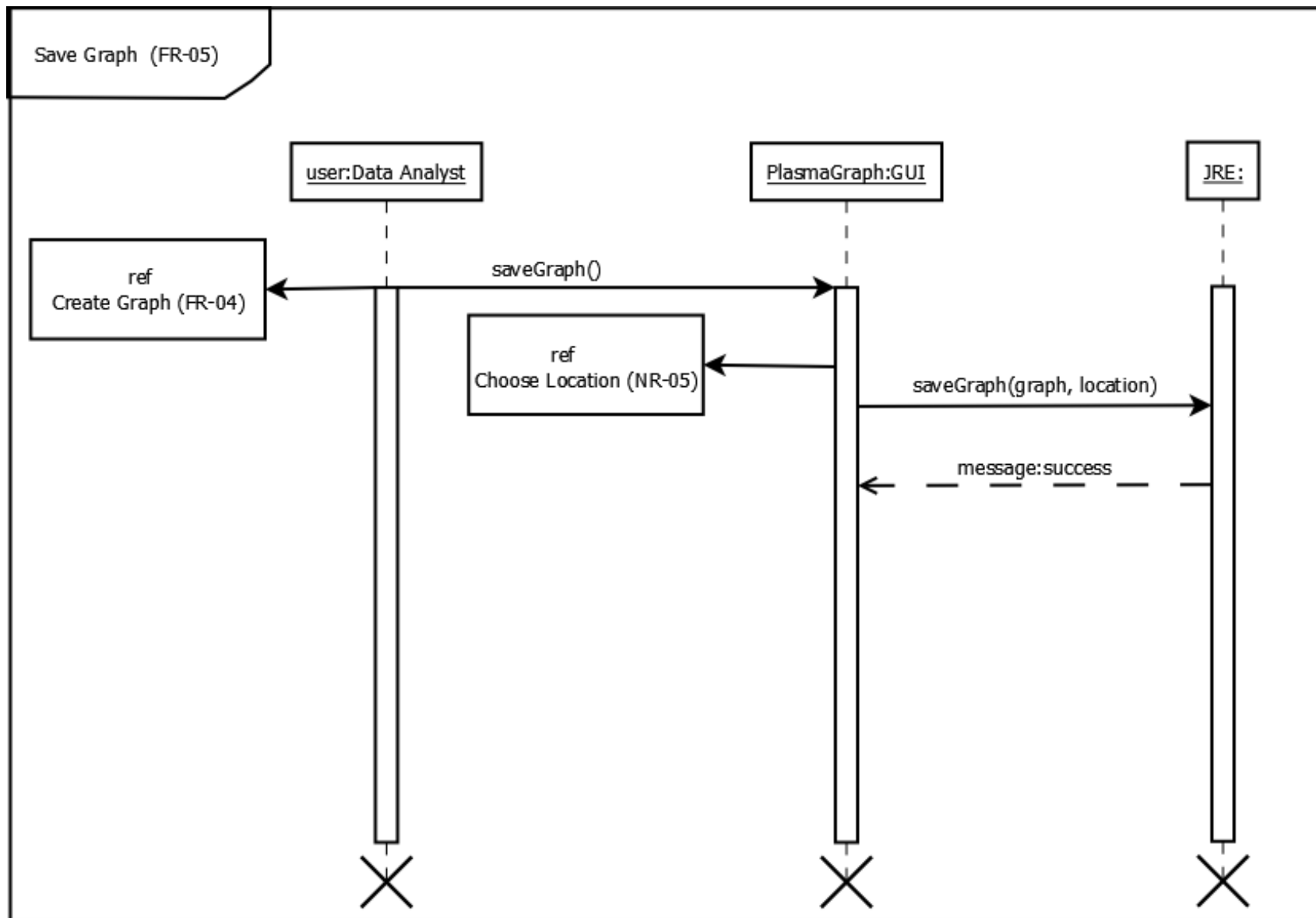


Figure 52: "Save Graph" Sequence Diagram



3.2.3.2. Non-Functional Requirements

Figure 53: “Save Template” Sequence Diagram

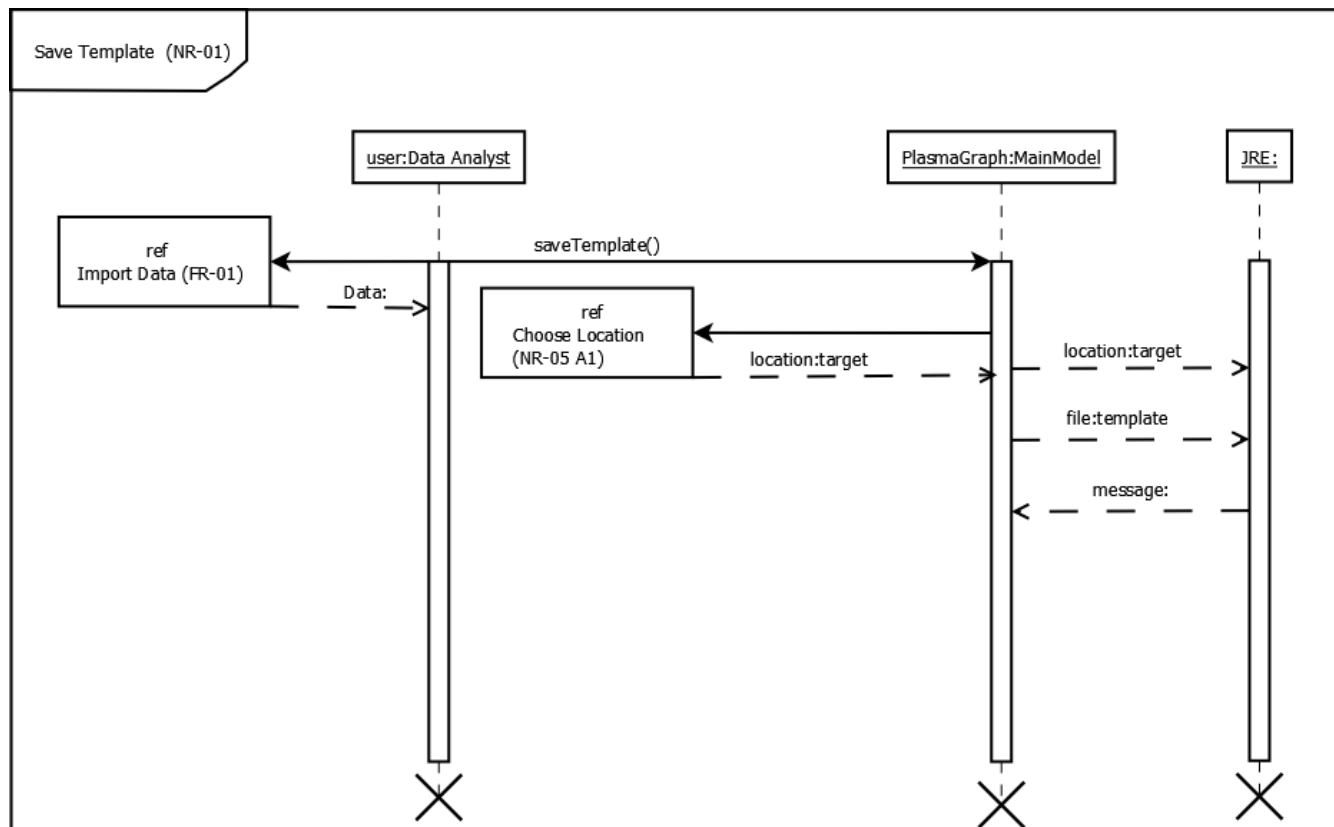


Figure 54: “Import Template” Sequence Diagram

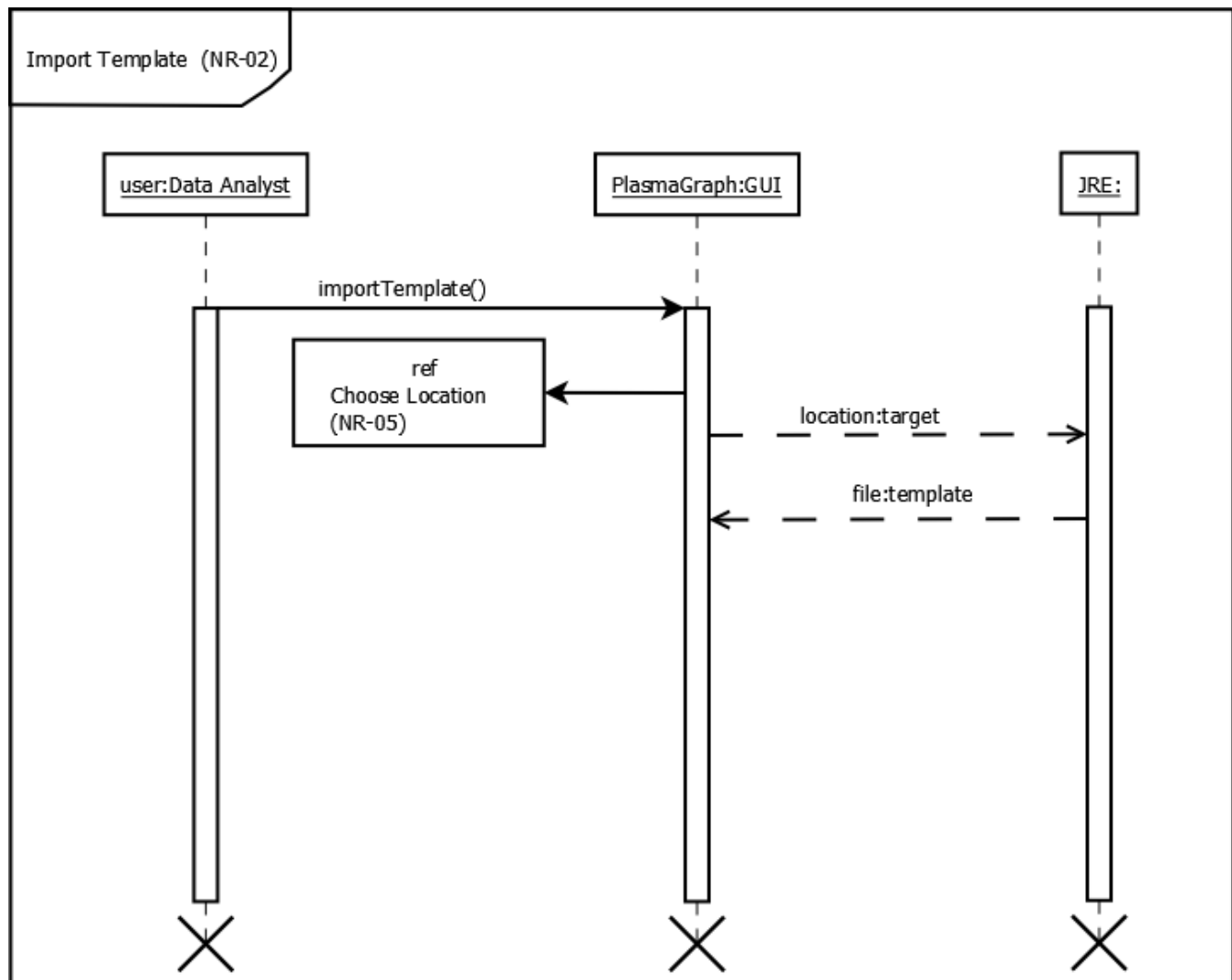


Figure 55: “Inspect Data” Sequence Diagram

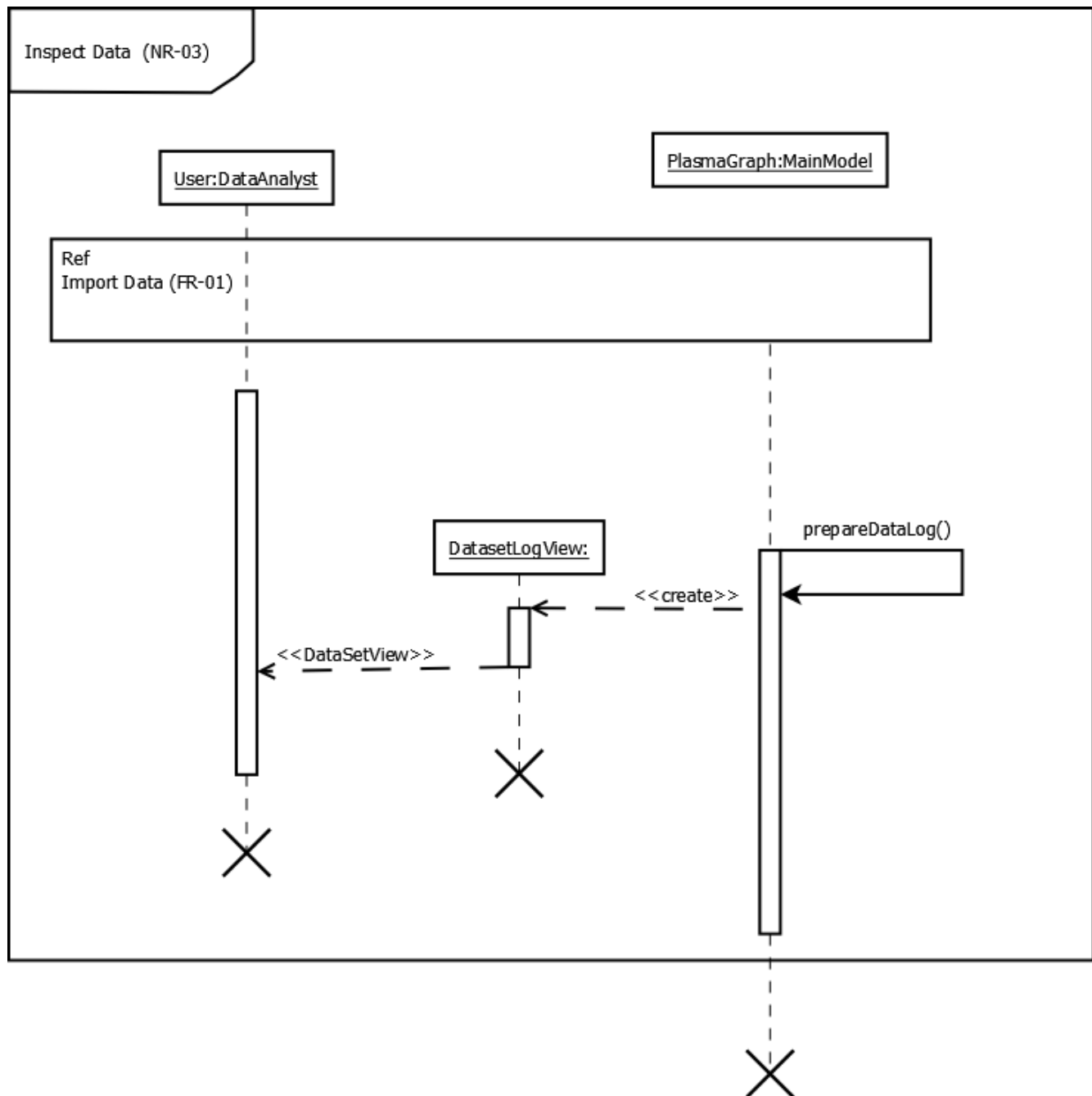


Figure 56: “Display Help” Sequence Diagram

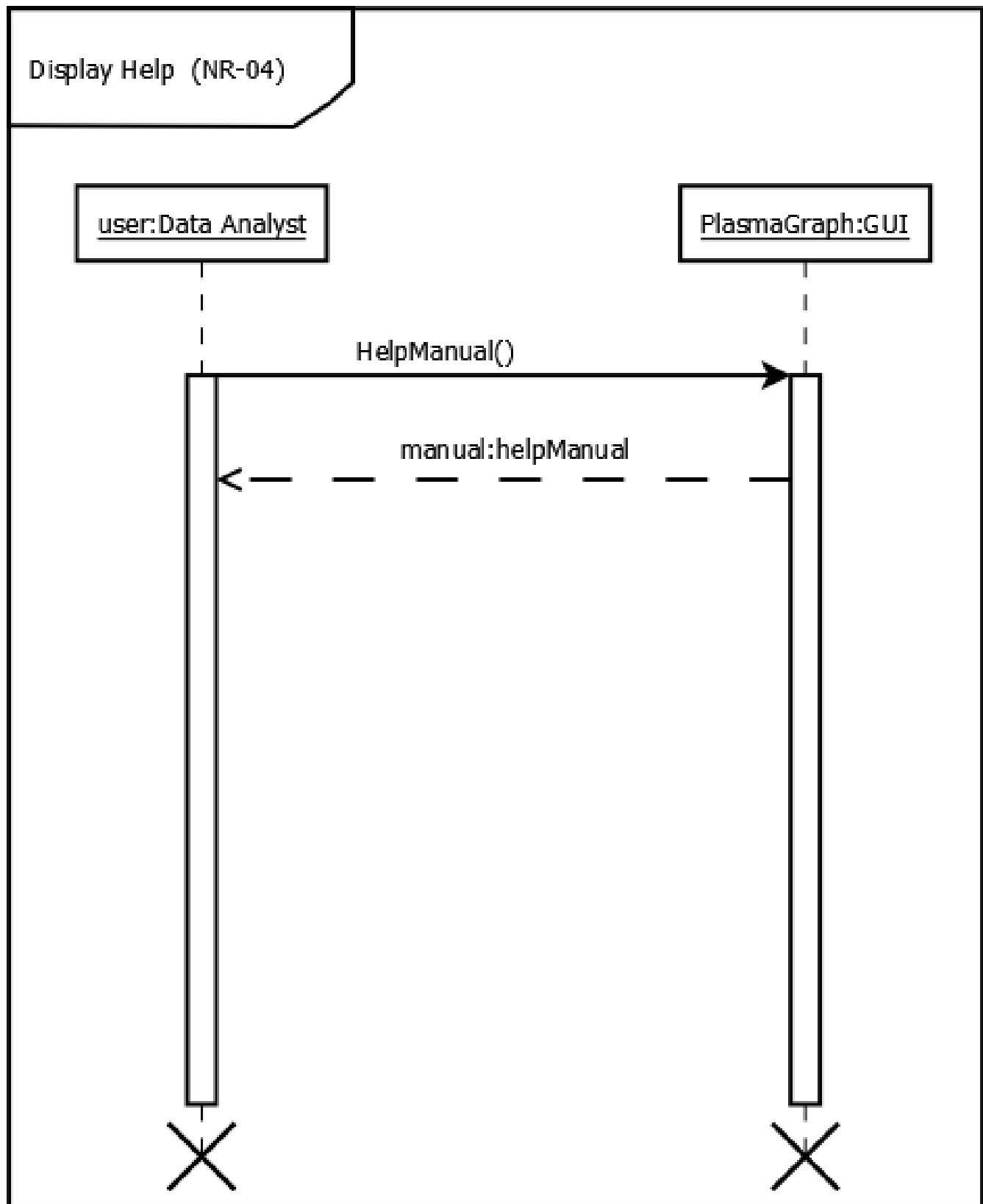


Figure 57: “Choose Location” Sequence Diagram

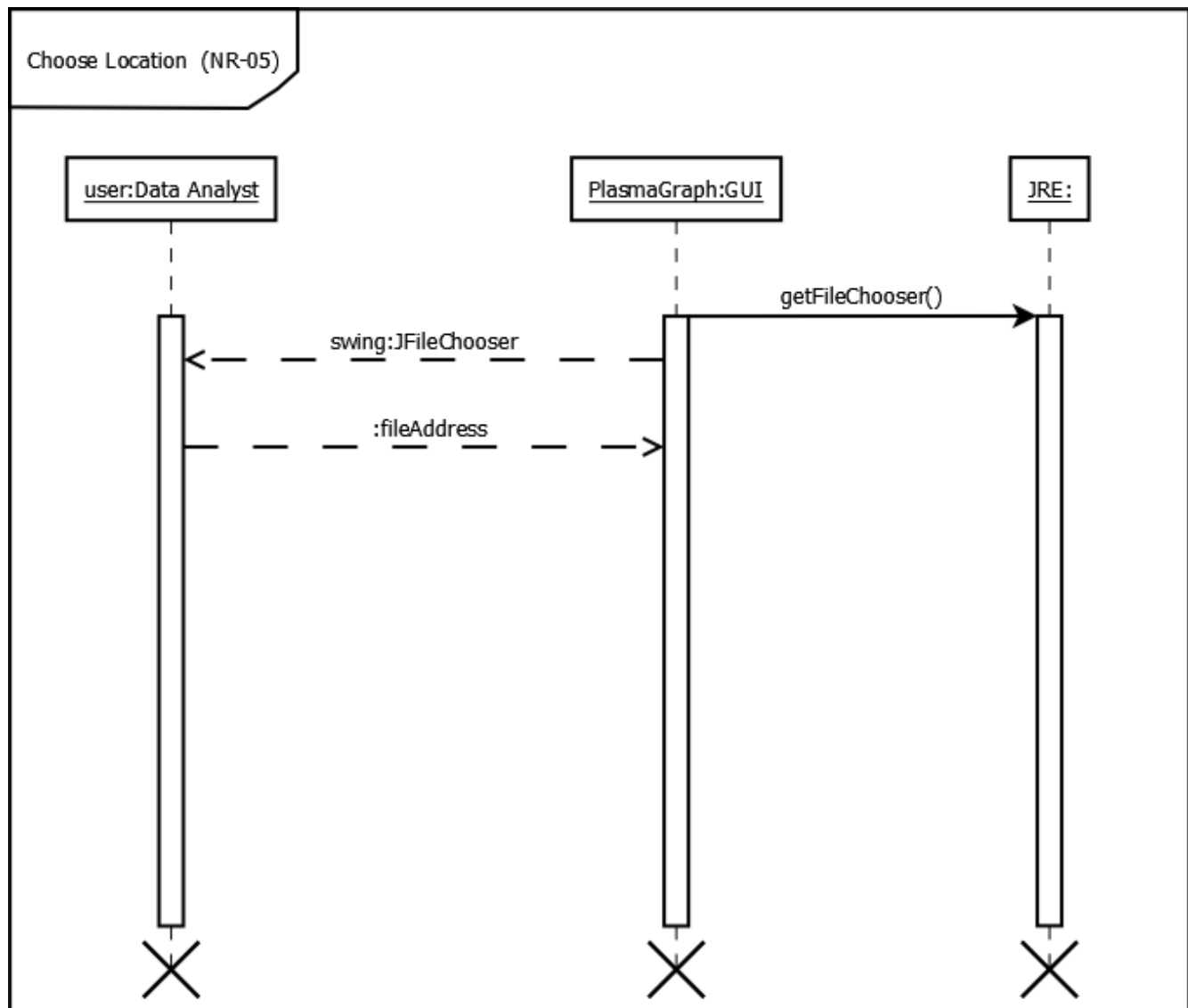
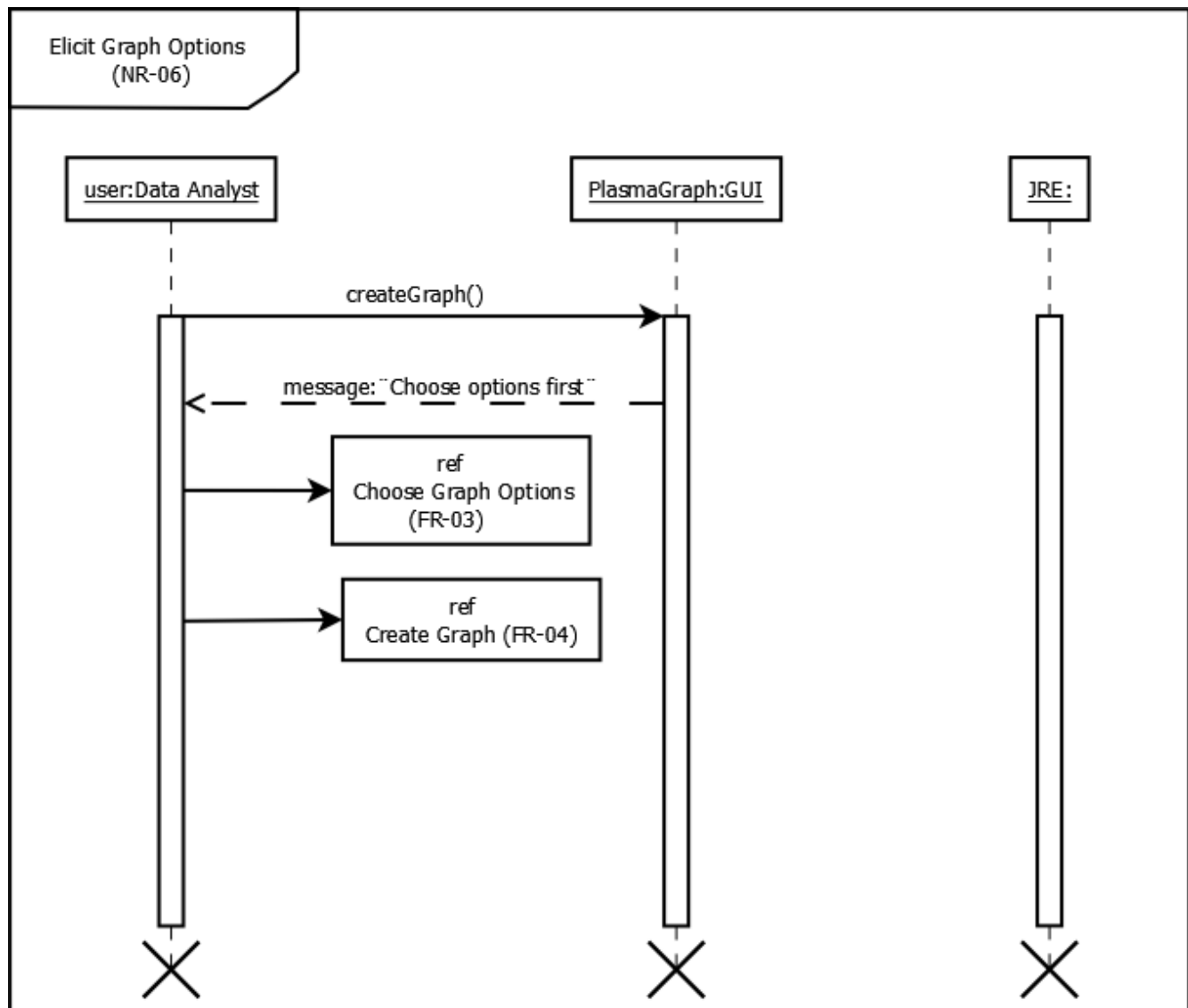


Figure 58: “Elicit Graph Options” Sequence Diagram



3.3 Design Rationale

PlasmaGraph has undergone many small changes from its initial design. The initial design's key visual features and their issues are as follows:

- The GUI separated visual column settings and the selection of columns to graph into different windows, forcing the user to switch between both windows to modify parts of the same graph.
- A user could not modify the same graph; instead, a new graph would be produced. This created confusion between the users regarding which was the most recent version of the graph.

These key design features were supported by various internal design decisions. The previous iteration's structure and the changes made are:

- The GUI was separated into multiple sections based on the different classes of settings available. Instead, all the settings were consolidated into one main window, and all tools (Interpolation and Outlier Scanning) were placed into another. This provided a clear separation between cosmetic and tool settings.
- The graphing procedure would create a new window for the graph each time the "Graph" button was pressed. Instead, a new window was made to contain the graph, and the "Graph" button was made to update the window with a new graph.

These changes allow the program to be focused on editing individual graphs at a time and provide clear instructions on how a user can modify a graph with the options available.

4. DATA DESIGN

The data objects that will form a part of PlasmaGraph are:

- HeaderData (Located in: org.pvg.plasmagraph.utils.data)
- GraphPair (Located in: org.pvg.plasmagraph.utils.data)
- DataSet (Located in: org.pvg.plasmagraph.utils.data)
- Template (Located in: org.pvg.plasmagraph.utils.template)

These objects contain the data and settings required by PlasmaGraph in order to produce a graph.

The following describes the purpose of each object in relation to the program's requirements as stated in the SRS:

- HeaderData: This object holds the user-provided data file's location, as well as a list of the data file's column names and column data types. This component can be combined with a GraphPair object to create a DataSet. The HeaderData object is part of the functional requirement FR-01.
- GraphPair: This object contains the index references of the X, Y, and Group Columns, as well as their names. The GraphPair is a main component of all graphs, and allows for the easy creation of DataSets via its HeaderData column references. The GraphPair object is part of the functional requirement FR-03 and FR-04.
- DataSet: The DataSet object is the synthesis of the HeaderData and GraphPair objects, containing the X, Y, and (if selected) Group Columns' data. This object is used in conjunction with the Template to produce graphs. The DataSet object is part of the functional requirement FR-03 and FR-04.
- Template: The Template object is a container of any graph settings that are not derived from the data file itself. It is one of the two indispensable components (along with the GraphPair) of any PlasmaGraph graph. The Template object is part of the functional requirement FR-03.

Section 3.2 of this document details the underlying composition of these objects, and Section 5 details the processes by which these objects are combined to create graphs.

5. COMPONENT DESIGN

The following section will describe all class functions vital to the furthering of the features listed in Section 2.1:

5.1. PlasmaGraph

5.1.1. main ()

void main (String [] args):

- Start a new java process with the following:

 - Initialize the Template, HeaderData, and GraphPair objects.

 - Initialize the Graph MVC objects.

 - Initialize the Data Set MVC objects.

 - Initialize the Tool MVC objects.

 - Initialize the Main MVC objects.

- Set the main and graph views to be visible.

- Ask the user for a data file to use.

5.2. MainModel

5.2.1. importData ()

void importData ():

- Open a File Choosing Window and allow the user to select a file.

- If the File Chooser Window was closed via the "Open" button:

 - If the file selected is a Matlab file:

 - Read the file's column names and types into the HeaderData object.

 - Notify all GUI components of changes done to the HeaderData object.

 - Reset the GraphPair object.

 - Otherwise:

 - Open an Error Window stating this program cannot read that type of file.

- Otherwise:

 - Open an Error Window stating that the operation was cancelled.

5.2.2. importTemplate ()

void importTemplate ():

- Open a File Choosing Window and allow the user to select a file

- If the File Chooser Window was closed via the "Open" button:

- If the file selected is a Template file:.

- Ask the user if they want to overwrite their current settings.

- If the user responds that they do:

- Open the template and put its contents into the Template object.

- Otherwise:

- Open an Error Window stating that the Template will not be opened.

- Otherwise:

- Open an Error Window stating this program cannot read that type of file.

- Otherwise:

- Open an Error Window stating that the operation was cancelled.

5.2.3. saveTemplate ()

void saveTemplate ():

- Open a File Choosing Window and allow the user to select a location and name for the file.

- If the File Chooser Window was closed via the "Open" button:

- Save the file under the selected name in the selected location.

- Otherwise:

- Open an Error Window stating that the operation was cancelled.

5.2.4. prepareDataLog ()

void prepareDataLog ():

- If the file in the HeaderData object is a Matlab file:

- Open the file and obtain the text representation of the data within the file.

- Open a DatasetLogView window with the extracted text.

- Make the DatasetLogView window visible.

- Otherwise:

- Open a DatasetLogView window with no text.

- Make the DatasetLogView window visible.

5.3. GraphModel

5.3.1. graph ()

JFreeChart graph ():

If the user has selected to use the Outlier Search feature:

Create a new DataSet object.

Perform the Outlier Search on the HeaderData object columns dictated by the GraphPair object.

Store the Outlier Search results in the DataSet object.

If the user has selected to use the Interpolation feature:

Return an interpolated XYGraph object using the DataSet, Template, and GraphPair objects.

Otherwise:

Return an XYGraph object using the DataSet, Template, and GraphPair objects.

Otherwise:

If the user has selected to use the Interpolation feature:

Return an interpolated XYGraph object using the HeaderData, Template, and GraphPair objects.

Otherwise:

Return an XYGraph object using the HeaderData, Template, and GraphPair objects.

5.4. GraphView

5.4.1. graphUpdate ()

void graphUpdate ():

If the HeaderData object is empty:

Show an error window explaining that the Import Data procedure must be successfully completed before being able to graph.

Show an empty graph in the GraphView window.

If the HeaderData object does not contain valid column data types:

Show an error window explaining that the selected columns must be of the proper type for the selected chart type.

Show an empty graph in the GraphView window.

If the GraphPair object is not ready to be graphed:

Show an error window explaining that the X and Y Columns must be selected before graphing.

Show an empty graph in the GraphView window.

If the X and Y Columns to be graphed from the HeaderData object are the same column:

 Show an error window explaining that the X and Y Columns cannot be the same column.

 Show an empty graph in the GraphView window.

If there is a valid Group Column:

 If the X and Y Columns to be graphed from the HeaderData object are the same as the Group Column:

 Show an error window explaining that the X and Y Columns cannot be the same as the Group Column.

 Show an empty graph in the GraphView window.

Otherwise:

 If the user has selected to perform the Outlier Search function:

 Show a graph of the data without the scan.

 Show a graph of the data using the HeaderData, Template, and GraphPair objects.

5.5. HeaderData

5.5.1. populateData (GraphPair p)

DataSet populateData (GraphPair p):

 Prepare a new DataSet object using the GraphPair object.

 Read the data file in the HeaderData object.

 Place the contents of the columns specified in the GraphPair object into the DataSet object.

5.6. DataSet

5.6.1. toXYGraphDataset (GraphPair p)

XYSeries toXYGraphDataset (GraphPair p):

 Create a new XYSeries object with the name provided by the GraphPair object.

 For each row in this DataSet object:

 Add the row's X and Y values into the XYSeries object.

 Return the XYSeries object.

5.6.2. toGroupedXYGraphDataset (GraphPair p)

XYSeriesCollection toGroupedXYGraphDataset (GraphPair p):

- Create a HashMap object containing Object and XYSeries objects.

- For each row in this DataSet object:

 - If the HashMap object contains a value equal to the Group Column's value in the row:

 - Add the row's X and Y values into that XYSeries object.

 - Otherwise:

 - Create a new XYSeries object with that Group Column value.

 - Add the row's X and Y values into that XYSeries object.

 - Add that Group Column value and XYSeries object into the HashMap.

- Create a new XYSeriesCollection object.

- For each XYSeries object in the HashMap:

 - Add the XYSeries object into the XYSeriesCollection object.

- Return the XYSeriesCollection object.

5.7. MatlabProcessor

5.7.1. getHeaders (HeaderData hd)

boolean getHeaders (HeaderData hd):

- If the file hasn't been read yet:

 - Read the data file's contents

 - Store its contents within this MatlabProcessor object's HashMap.

- For each MLArray object in the HashMap:

 - If the MLArray is empty or only contains null or NaN values:

 - Remove that MLArray from the HashMap.

- If the HashMap contains two or more MLArrays:

 - If each MLArray in the HashMap contains the same number of rows:

 - For each MLArray in the HashMap:

 - Create a new HeaderComponent containing the MLArray's name and data type.

 - Add the HeaderComponent object to the HeaderData object.

 - Set the HeaderData object's file to that used in this MatlabProcessor object.

- Return true.

Otherwise:

Open an error window explaining that the data file contained MLArrays with different numbers of rows.

Otherwise:

Open an error window explaining that the data file contained less than two graphable columns and cannot be used.

5.7.2. toDataSet (DataSet ds, GraphPair p, HeaderData hd)

void toDataSet (DataSet ds, GraphPair p, HeaderData hd):

If the file hasn't been read yet:

Read the data file's contents.

Store its contents within this MatlabProcessor object's HashMap.

Create a list object.

For each MLArray in this MatlabProcessor object's HashMap:

If the MLArray's name is the same as that of a column in the GraphPair object:

Add that MLArray to the list object.

For each row in the MLArrays in the list:

If the row only contains valid values:

Add that row's X Column MLArray value to the DataSet object's X Column.

Add that row's Y Column MLArray value to the DataSet object's Y Column.

If the GraphPair contains a Group column:

Add that row's Group Column MLArray value to the DataSet object's Group Column.

5.8. Template

5.8.1. saveTemplate (String s)

void saveTemplate (String s):

Create a Writer object containing the File object to write to.

Open the File object via the Writer object.

Write the contents of the Chart Type variable to the File.
Write the contents of the Chart Name variable to the File.
Write the contents of the X Axis Label variable to the File.
Write the contents of the Y Axis Label variable to the File.
Write the contents of the Using Legend variable to the File.
Write the contents of the Using Tooltips variable to the File.
Write the contents of the Generate URLs variable to the File.
Write the contents of the Orientation variable to the File.
Write the contents of the X Axis Type variable to the File.
Write the contents of the Y Axis Type variable to the File.
Write the contents of the Default Interpolation Type variable to the File.
Write the contents of the Default Outlier Reaction variable to the File.
Write the contents of the Interpolation Lower Range variable to the File.
Write the contents of the Interpolation Upper Range variable to the File.
Write the contents of the Interpolation Point Amount variable to the File.
Write the contents of the Outlier Distance variable to the File.
Write the contents of the Outlier Distance Type variable to the File.

5.8.2. `openTemplate` (File f)

`void openTemplate (File f):`

Create a Reader object containing the File object to read from.
Open the File object via the Reader object.

Obtain a line of text from the File object.
Place its contents into this Template object's Chart Type variable.

Obtain a line of text from the File object.
Place its contents into this Template object's Chart Name variable.
Obtain a line of text from the File object.
Place its contents into this Template object's X Axis Label variable.
Obtain a line of text from the File object.
Place its contents into this Template object's Y Axis Label variable.

Obtain a line of text from the File object.
Place its contents into this Template object's Using Legend variable.
Obtain a line of text from the File object.
Place its contents into this Template object's Using Tooltips variable.
Obtain a line of text from the File object.
Place its contents into this Template object's Generate URLs variable.

Obtain a line of text from the File object.

Place its contents into this Template object's Orientation variable.

Obtain a line of text from the File object.

Place its contents into this Template object's X Axis Type variable.

Obtain a line of text from the File object.

Place its contents into this Template object's Y Axis Type variable.

Obtain a line of text from the File object.

Place its contents into this Template object's Default Interpolation Type variable.

Obtain a line of text from the File object.

Place its contents into this Template object's Default Outlier Reaction variable.

Obtain a line of text from the File object.

Place its contents into this Template object's Interpolation Lower Range variable.

Obtain a line of text from the File object

Place its contents into this Template object's Interpolation Upper Range variable.

Obtain a line of text from the File object

Place its contents into this Template object's Interpolation Point Amount variable.

Obtain a line of text from the File object

Place its contents into this Template object's Outlier Distance variable.

Obtain a line of text from the File object

Place its contents into this Template object's Outlier Distance Type variable.

Notify all windows about changes in this object.

5.9. Interpolator

5.9.1. interpolate ()

Graph interpolate ():

If the Interpolation object's GraphPair contains a Group column:

Convert the Interpolation object's DataSet to a DataSet XYSeriesCollection object.

Create a Storage XYSeriesCollection object.

For each XYSeries object in the DataSet XYSeriesCollection:

Create the interpolated XYSeries object of the Collection's XYSeries object based on the Template's Interpolation Type.

Insert the interpolated XYSeries object into the storage XYSeriesCollection.

For each XYSeries object in the storage XYSeriesCollection:

If the object is not an empty XYSeries object:

Add the XYSeries to the DataSet XYSeriesCollection object.

Return an XYGraph containing the Template the DataSet XYSeriesCollection, and the GraphPair.

Otherwise:

Convert the Interpolation object's DataSet to a DataSet XYSeries object.

Create an interpolation of the DataSet XYSeries Object based on the Template's Interpolation Type.

Create an XYSeriesCollection containing the DataSet XYSeries object and the interpolation XYSeries object.

Return an XYGraph containing the Template, the XYSeriesCollection, and the GraphPair.

5.9.2. getInterpolation (XYSeries s)

XYSeries getInterpolation (XYSeries s):

If the user requests a linear interpolation:

Create a SimpleRegression object containing the XYSeries data.

Create a linear regression with the SimpleRegression object.

Return an XYSeries containing the linear regression data.

If the user requests a quadratic interpolation:

Create a rank 2 polynomial regression via the Regression object's "getPolynomialRegression" function.

Return an XYSeries containing the data within the polynomial regression.

If the user requests a cubic interpolation:

- Create a rank 3 polynomial regression via the Regression object's "getPolynomialRegression" function.

- Return an XYSeries containing the data within the polynomial regression.

If the user requests a spline interpolation:

- Order the XYSeries data in ascending X Column order.

- Create a SplineInterpolator object.

- Create a PolynomialSplineFunction object with the X and Y Column data.

- Return an XYSeries containing the data within the PolynomialSplineFunction.

5.10. ClusterScanning

5.10.1. scan (HeaderData hd, Template t, GraphPair p)

DataSet scan (HeaderData hd, Template t, GraphPair p):

- Obtain the DataSet object from the HeaderData and GraphPair objects.

If the Template object's Outlier Distance Type is equal to the Mahalanobis setting:

- Create a new MahalanobisDistance object.

Otherwise:

- Create a new CartesianDistance object whose distance is equal to the Template object's Outlier Distance value.

Obtain a List of clustered data points via the DBSCANClusterer object.

If the Template object's Outlier Response is to warn the user:

- Show a window describing which points are considered outliers based on the Template object's Outlier Distance Type.

Ask the user if they wish to remove the outlier points.

If the user affirms:

- Return a DataSet containing the largest cluster of points provided by the DBSCANClusterer object.

6. HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

PlasmaGraph's Graphical User Interface is designed to provide the user the tools to modify a single graph at a time. The program (which can be seen in Section 6.2) is divided into two windows:

- The Settings Window (Located on the left side of the screen): This window shows the available options to the user. This window also contains a Menu Bar that provides various data-related functions and a “Graph” button that triggers an update in the Graph Window. The Menu Bar contains the following:
 - “Data” Menu Bar Item: This menu provides options related to the data currently in use by PlasmaGraph.
 - “Import Data” Menu Item: Allows the user to select a new data file to use.
 - “View Data” Menu Item: Allows the user to view the data file currently in use.
 - “Templates” Menu Bar Item: This menu provides options related to the settings currently in use by PlasmaGraph.
 - “Import Template” Menu Item: Allows the user to import settings they had saved previously.
 - “Save Template” Menu Item: Allows the user to save settings they have selected for the current graph for later use.
 - “Help” Menu Bar Item: This menu provides options related to the User Guide manual included with PlasmaGraph.
 - “User Guide” Menu Item: Displays a manual that shows the various ways to use PlasmaGraph.

Furthermore, the Settings window is divided into two tabs:

- “Data View” Tab: Contains all the cosmetic settings as well as the X, Y, and Group Column settings. Refer to Section 6.3 for information regarding the individual settings in this tab.
- “Options View” Tab: Contains all the optional tool settings for the Interpolation and Outlier Scanning features. Refer to Section 6.3 for information regarding the individual settings in this tab.

- The Graph Window (Located on the right side of the screen): This window shows the graph and allows the user to save the graph via a right click on the graph itself. This window's graph updates itself whenever the "Graph" button is pressed on the Settings Window.

The program's basic use flow is as follows:

1. User opens program.
2. User selects a file to use via the Import Data file-choosing window that automatically appears when the program is started.
3. User selects any number of options in the Data View or Options View tabs.
4. User presses the "Graph" button on the bottom of either tab.

6.2 Screen Images

Figure 59: PlasmaGraph Graphical User Interface Windows

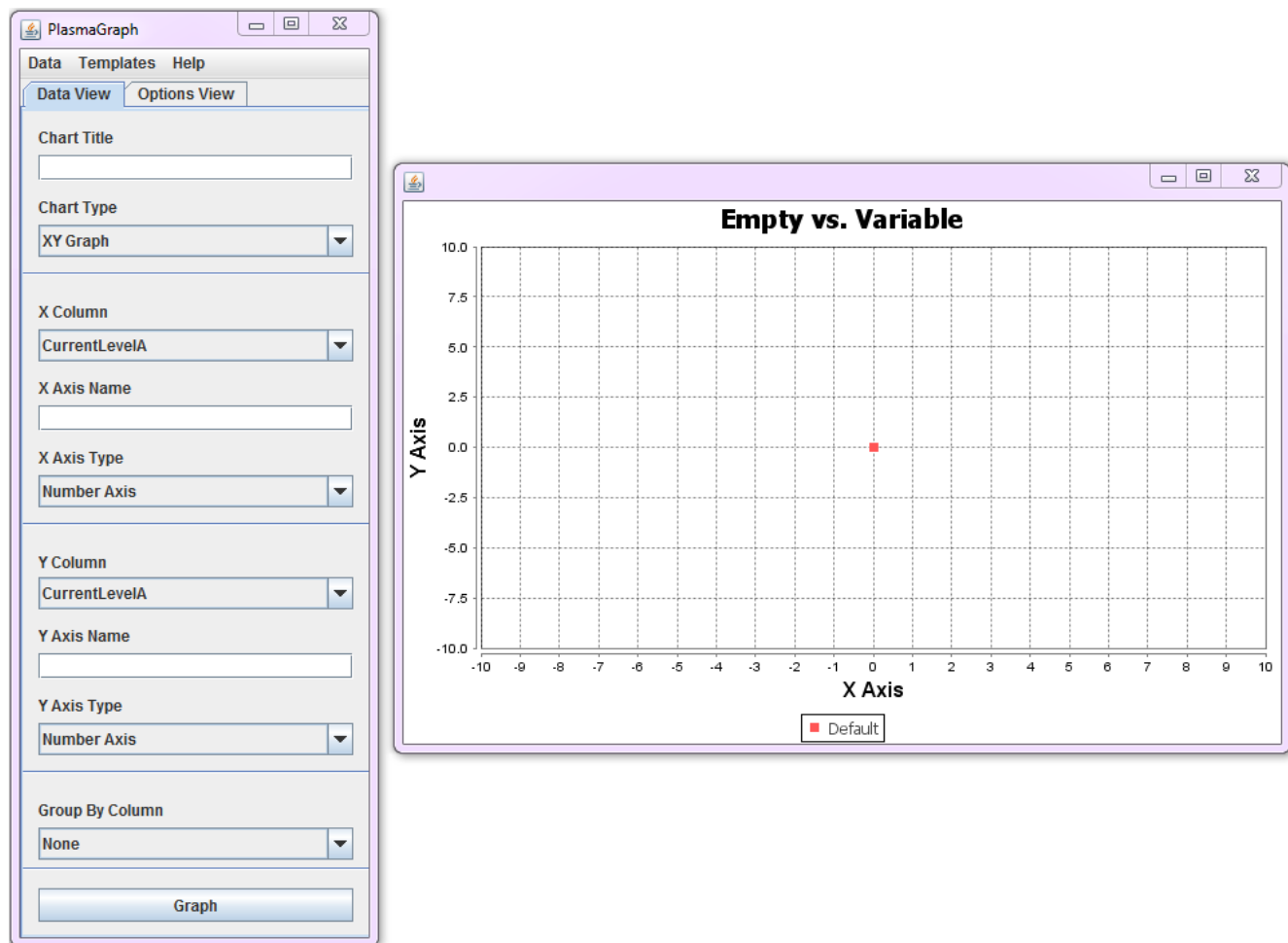


Figure 60: PlasmaGraph Settings Window Tabs

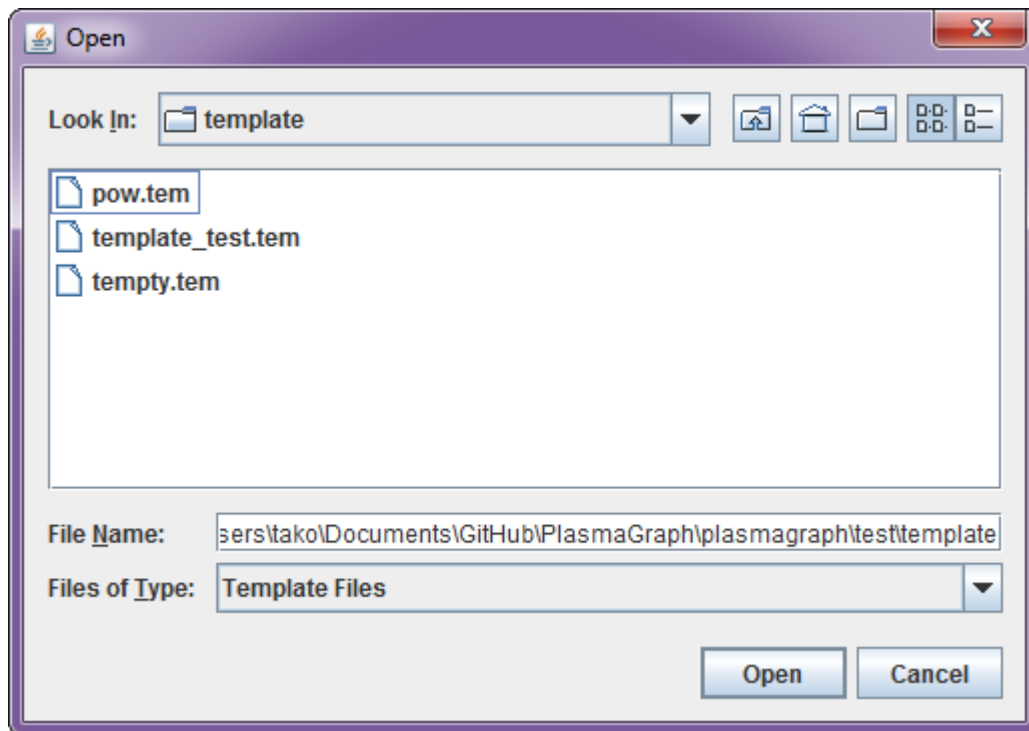
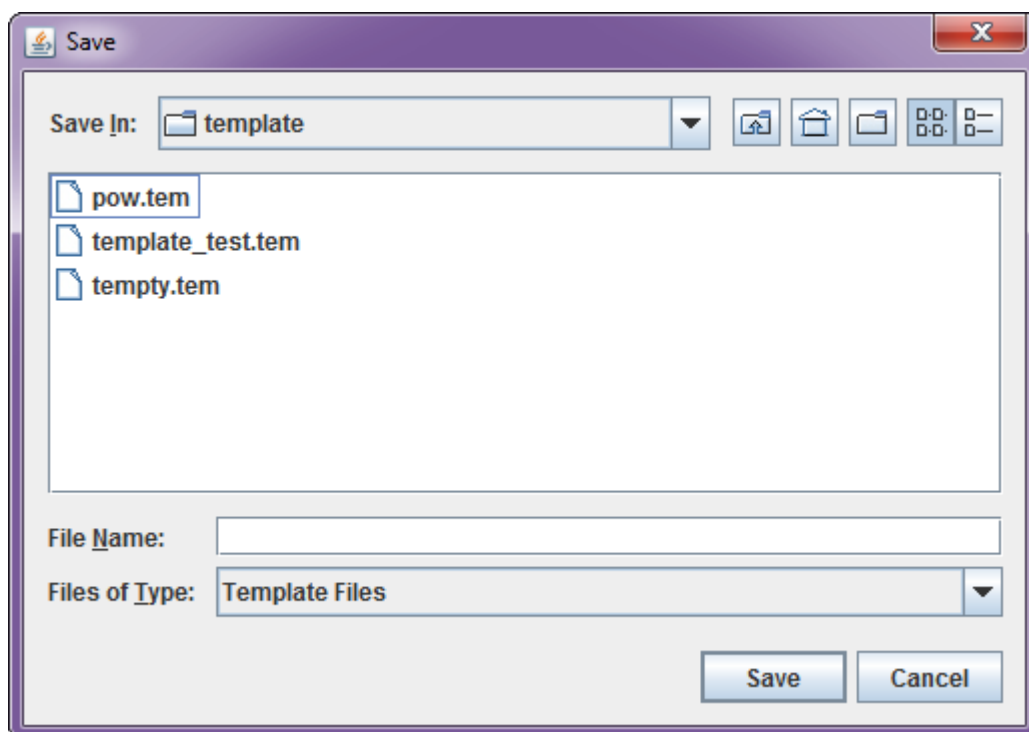
The figure displays two side-by-side screenshots of the PlasmaGraph Settings window, showing the 'Data View' and 'Options View' tabs.

Data View (Left Window):

- Chart Title:** Text input field.
- Chart Type:** Dropdown menu (XY Graph).
- X Column:** Dropdown menu (CurrentLevelA).
- X Axis Name:** Text input field.
- X Axis Type:** Dropdown menu (Number Axis).
- Y Column:** Dropdown menu (CurrentLevelA).
- Y Axis Name:** Text input field.
- Y Axis Type:** Dropdown menu (Number Axis).
- Group By Column:** Dropdown menu (None).
- Graph:** Button.

Options View (Right Window):

- Outlier Distance Type:** Dropdown menu (User-Provided Distance).
- Outlier Scanning Response:** Dropdown menu (No Scanning).
- Manual Maximum Distance:** Text input field (0.0).
- Interpolation Type:** Dropdown menu (None).
- Graph:** Button.

Figure 61: Open File / Template Window*Figure 62: Save Template Window*

6.3 Screen Objects and Actions

This section will discuss the various components of PlasmaGraph's GUI and what functionality they provide to the user.

- GUI Menu Bar
 - Refer to Section 6.1 for all components.
- GUI Body
 - Chart Title Text Box: Allows the user to provide a name to the graph. If left empty, PlasmaGraph will automatically populate the graph's Chart Title with the names of the X Column and Y Column Selection Box selections.
 - Chart Type Selection Box: Allows the user to select between various types of graphs. Currently, the only selection available to users is the "XY Graph" option, which provides a XY Scatter Plot, but, in future expansions, other types of graphs may be included.
 - X Column / Y Column Selection Box: Allows the user to select between columns in the imported data file. Currently, these Selection Boxes only display columns graphable by the selected Chart Type.
 - X Axis Name / Y Axis Name Text Box: Allows the user to provide a name to the X or Y Axes. If left empty, PlasmaGraph will automatically populate the graph's X and Y Axes Labels with the names of the X Column and Y Column Selection Box selections, respectively.
 - X Axis Type / Y Axis Type Selection Box: Allows the user to select between the available Axis scales for the graph. Currently, the options are:
 - Number Axis: The standard numerical scale.

- Logarithmic Axis: The standard logarithmic (base-10) scale.
- Group By Column Selection Box: Allows the user to select between columns in the imported data file for a third column to separate the data by. Selecting any option besides “None” creates clusters of X and Y Column values based on the different types of values in the Group By Column.
- Outlier Distance Type Selection Box: Allows the user to select the type of measuring mechanism to be used when scanning for outliers. Currently, the options for this Selection Box are:
 - User-Provided Distance: Allows the user to provide the Cartesian distance between points where points are statistically invalid.
 - Mahalanobis Distance: Utilizes a modified version of the standard deviation to determine whether points are statistically invalid.
- Outlier Scanning Response Selection Box: Allows the user to select whether Outlier Scanning will be performed. The “Warn” and “Remove” options both provide a window detailing the different values removed from the graph. “Remove” will automatically remove them, and “Warn” will provide the user the option to remove the potential outliers.
- Manual Maximum Distance Text Box: Allows the user to select the maximum Cartesian distance between points for the purposes of Outlier Scanning. It is only used when the Outlier Distance Type Selection Box is set to “User-Provided Distance”.
- Interpolation Type Text Box: Allows the user to select the type of interpolation to perform on all the data. The available options are:
 - None: No interpolation will occur.
 - Linear: The data will be interpolated linearly.
 - Quadratic: The data will be interpolated with a quadratic line.
 - Cubic: The data will be interpolated with a cubic line.
 - Spline: The data will be interpolated with a curve passing through each data point. For this option, the data must pass the Vertical Line Theorem (no two points may have the same X axis value).

