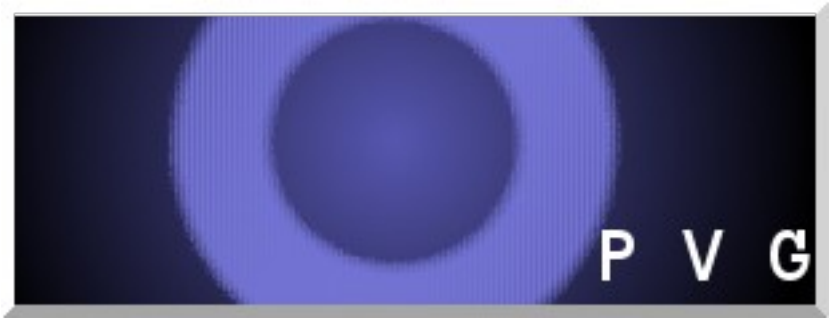
PlasmaGraph

Software Project Management Plan



By: Plasma Visualization Group

Polytechnic University of Puerto Rico

Electrical & Computer Engineering and Computer Science Department

Fall – Winter 2013

CS 4200: Computer Science Senior Project

Professor: Luis A. Ortiz

Version: SPMP r1

Members:

Daniel E. Quintini Greco [CS - # #####]

Gerardo A. Navas Morales [CS - # #####]

Revision Chart

- Friday, November 11, 2013 – r1 - First version of document.

Preface

Table of Contents

Section	Page
1. Introduction	-
1.1. Project Overview	-
1.2. Project Deliverables	-
1.3. Evolution of the SPMP	-
1.4. Reference Materials	-
1.5. Definitions and Acronyms	-
2. Project Organization	-
2.1. Process Model	-
2.2. Organizational Structure	-
2.3. Organizational Boundaries and Interfaces	-
2.4. Project Responsibilities	-
3. Managerial Process	-
3.1. Management Objectives and Priorities	-
3.2. Assumptions, Dependencies, and Constraints	-
3.3. Risk Management	-
3.4. Monitoring and Controlling Mechanisms	-
3.5. Staffing Plan	-
4. Technical Process	-
4.1. Methods, Tools, and Techniques	-
4.2. Software Documentation	-
4.3. Project Support Functions	-
5. Work Packages, Schedule, and Budget	-
5.1. Work Packages	-
5.2. Dependencies	-
5.3. Resource Requirements	-

5.4. Budget and Resource Allocation -
5.5. Schedule -

Additional Components -
Index -
Appendices -
A1. Gantt Chart -

1. Introduction

1.1. Project Overview

This document will cover the plan regarding the execution of the "PlasmaGraph" tool for the Polytechnic University of Puerto Rico's Plasma Laboratory. Said product, "PlasmaGraph" is an intermediary tool, serving as a channel to funnel data provided by the Plasma Laboratory's main machine to a work or personal computer in the form of any kind of graph the user(s) wish to create. Said graphs will be able to be saved to the computer as well as displayed directly via the computer screen. The final objective is a tool that can provide graphs based on arbitrarily-large CSV (Comma-separated value) files that may have some errors.

As the project is rather small, its incremental goals are also relatively small. PlasmaGraph requires a backbone composed of a storage mechanism and a parsing mechanism to place the CSV data into said storage before being able to graph the data. Upon completion of this system, work on the error-correction system, graphing system, and GUI (Graphical User Interface) can begin in earnest, though not necessarily in that exact order. Said milestones will be tested via PUPR Plasma Laboratory data and dummy data designed to simulate certain errors that the client has explained.

Two current Computer Science students studying in the Polytechnic University of Puerto Rico will work to complete this project, alongside a university-provided mentor. They will use both their own computing resources and those of the university (to test the product's speed and reliability in various states of completion), as well as actual laboratory data in order to complete this project. The cost of completing this project, as explained later in this document, will be primarily from the labor of the students and the mentor. Completion of this project will take approximately 3 months; the exact composition of this schedule will be explained later in this document, as will the budgetary and resource requirements.

As an aside, this project does not connect to any currently-active project, though it does connect to a physical system currently in use; the PUPR Plasma Laboratory is active as of the time of this project and documentation.

1.2. Project Deliverables

The client, Professor Angel Gonzales-Lizardo of the Polytechnic University of Puerto Rico's Plasma Laboratory, must receive the following items at his office in the PUPR Plasma Lab by <Enter Date Here> at <Enter Time Here> AST:

A. One (1) digital copy of the PlasmaGraph product, program documentation, and user manual in a <Insert Physical Medium Here>.

B. One (1) physical (paper, softcover) copy of the PlasmaGraph User Manual.

Additionally, the following will be provided to the Senior Project Professor, Professor Luis A. Ortiz:

A. One (1) physical copy of the following specification documents:

1. The Software Project Management Plan (SPMP)
2. The Software Requirements Specification (SRS)
3. The Software Design Description (SDD)
4. The Software Test Document (STD)

B. One (1) physical copy of all meeting notes that were held between the Senior Project Professor and the group.

C. One (1) digital copy of the PlasmaGraph product, program documentation, and user manual in a <Insert Physical Medium Here>.

D. One (1) physical (paper, softcover) copy of the PlasmaGraph User Manual.

Said copy will be used to evaluate the quality of the work performed in the project for the purpose of giving a final grade and confirming successful completion of the Computer Science course "Computer Science Senior Project".

As per the nature of how this project is being hosted, all documents will be also available on GitHub

(https://github.com/CherimaeNemeta/PlasmaGraph), including the specification documents.

1.3. Evolution of the SPMP

The SPMP will only be changed by the Project Lead.

The first version of this document, as every subsequent version, will be made in parts. Every party that composes the SPMP will be created and modified on its own, as an individual sub-document. When changes are finalized, all the parts will be compiled into one document and given a version number based on the previous version document's number; said number will start at "1" and progress incrementally. (The first version number will be "1"; this number will increase to "2" and so on.) Said version number will be prefaced in the name of the file by the letter "r". (As per the word "revision".) Therefore, the first version of the SPMP will be named "Software Program Management Plan r1.pdf", and the following versions will only have it's version number increase.

All components of this project related to the creation of the SPMP will be located in the project's GitHub version control system. (https://github.com/CherimaeNemeta/PlasmaGraph) A section of the repository will be dedicated only to the SPMP's fragments (specs/SPMP/Partes Incompletas/), and another will be for completed documents. (specs/Documentos Completos/SPMP/) Document fragments will be divided by the section they are located in (Folders labelled Sections 1 through 5 in said subdirectory.), and completed documents will be divided by version. (Folders labelled either "Current Version" or "Previous Versions" in said subdirectory.) Said system will manage the myriad files that the project will require and allow the project to recover from catastrophic losses of data, if and when they occur.

1.4. Reference Materials

All documents listed below are found in the project repository.
(https://github.com/CherimaeNemeta/PlasmaGraph)

Specification Document Manuals (IEEE) (specs/_Manuales de Software Engineering/)

1. IEEE 830 SRS 1998 ing.pdf
2. IEEE 1016 SDD 1998 ing.pdf
3. IEEE 829 STD 1998 ing.pdf

4. IEEE 1058 SPMP 1998 ing.pdf

Meeting / Report Document Templates

1. Reunion Template.txt (specs/Notas de Reuniones/Reuniones con Prof. Ortiz/)

1.5. Definitions and Acronyms

A. Definitions

- PlasmaGraph; Product: a CSV-File error-checking grapher made specifically for the PUPR Plasma Laboratory.
- Client: Professor Angel Gonzales-Lizardo; Person responsible for the request of this product.
- Senior Project Professor: Professor Luis A. Ortiz; Overseer of the project's progress.
- JFreeChart: A set of tools written in the programming language "Java" that create graphical representations of data provided to it.
- Java: Programming language relied on for its portability between multiple types of computers and flexible capabilities.
- Project Group; Group: The creators and maintainers of the project and all its end products. Composed of Gerardo A. Navas Morales, Daniel E. Quintini Greco, and Professor Luis A. Ortiz.
- GitHub: File repository system that uses the Git VCS.
- VCS: Version Control System, also known as Revision Control or Source Control. System that manages multiple users accessing and changing the same files without losing any data.

B. Acronyms

- PUPR: Polytechnic University of Puerto Rico; university where the Plasma Laboratory is located, as well as where the group's members study.
- SPMP: Software Project Management Plan; This document.
- SRS: Software Resource Specifications
- SDD: Software Design Documents

- STD: Software Testing Documentation
- CSV: Comma-Separated Values; a type of computer file characterized by data separated by commas and terminated by an end-of-line character, with the first set of data usually representing the table column header.
- GUI: Graphical User Interface.
- PNG: Portable Network Graphics; a type of computer file that holds an image.
- JPG / JPEG: Joint Photographic Expert Group; another type of computer file that holds an image.

2. Project Organization

2.1. Process Model

This project's completion will be measured via the milestones presented in Appendix A. Said milestones include document and product completion, as well as the necessary miscellaneous events that must be organized and vetted by the university to prove completion of the Computer Science Senior Project. The milestone documents include not only the four design documents (SRS, SDD, STD, SPMP), but also all reference documentation created for the product. (Chief of which is the Java Documentation that will be generated from our code documentation.) Product completion will be measured by the completion of the various modules that will compose the product, as per the Program Flow diagram presented in <?>.

Started: Wednesday, August 21, 2013 4:00 PM-4:30 PM (UTC-04:00)
End: Approximately December 5, 2013. (Tentative.)
Why not before FA-13 Trimester end?: Classes, work, etc. (Other obligations.) Also, buffer time for any necessary additional features that creep up suddenly, testing time, data cleanup tuning time, and other miscellaneous things that may require additional time.

Project Deliverables will be completed once steps <?> are done. Once this occurs, documents will be combined into their final form, the product will be compiled one last time and tested, and, upon the success of the final tests, the final product's presentation will be organized and delivered on <?>. Furthermore, the product will also be provided to both the client and the class professor (Luis A. Ortiz), along with all relevant documentation requested by both parties.

For additional specifics on the schedule for this project, please refer to the PlasmaGraph Gantt Chart provided in Appendix B of this document, which presents a schedule of this project's relevant milestones.

2.2. Organizational Structure

This project will be completed by two members: Gerardo A. Navas and Daniel Quintini. Each of these group members will have certain duties to fulfill that will overlap, as well as some that will be unique to each of them. Daniel Quintini will take the role of Design Lead, whereas Gerardo A. Navas will act as Project Lead. These two roles will be explained in depth in the following sections. Despite the names, they are both equivalent positions with regards to project management hierarchy.

2.3. Organizational Boundaries and Interfaces

The project group will handle the following items related to the project:

a. Creation and maintenance of the SRS, SDD, STD, and SPMP documents.
b. Research and development of the project's product.
c. Presentation of the product.
d. Meetings with both the client and the professor in charge.
e. Compilation and transferral of all project components to client.

This project group will perform only what is necessary to fulfill the requirements for the product.

2.4. Project Responsibilities

As per section 2.2 of this document, this project's group is composed of 2 members. The duties that these two members must perform either fall in two categories: those that will be shared and those that will not be shared between the two.

(Look at 2.3 for this one, and take those goals and dig down into the duties that will make them possible.)
The following are those duties that will overlap:
1. Research, design, develop, and maintain product.
2. Develop product documentation.
3. Develop and maintain SDD document.
4. Present weekly work update to Prof. Ortiz.
5. Schedule meetings with client.

6. Develop and present product presentation when completed.
7. Present all materials to Prof. Ortiz and Client.

The following are those duties that do not overlap:

Gerardo A. Navas Morales

1. Develop and maintain the SPMP and STD documents.
2. Maintain GitHub information repositories.
3. Compilation of all project materials when completed.

Daniel E. Quintini Greco

1. Develop and maintain SRS document.
2. Create various external interaction diagrams.

3. Managerial Process

3.1. Management Objectives and Priorities

This section will cover what the project group's goals are and how it will reach them.

A. Purpose

This project group was created in order to create a product for the client. Said project, PlasmaGraph, will be able to take, as input, a set of CSV-formatted data and the particulars of the graph (settings) the user wants created, and will provide, as output, the data provided as input in the form of the graph specified. It (the product) will also allow for said graph settings to be saved and used as desired.

B. Priority

The project is divided into two general sections due to the kind of work requested: the Program (The working product, including its JavaDoc files.) and its Documentation (This includes its specifications and the user manual that will be provided alongside the product.). We will allocate priority to the sections as follows.

1. Product Priority: The product will await development until both the requirements and design of the product have been confirmed and frozen. This means that the development of the product itself will be of a lower priority than the three primary specification documents (The SRS, SPMP, and SDD.), but it will be of equal priority to the STD and User Manual. (See B.2., Documentation Priority for more details regarding the delayed priority for the STD and User Manual.)

2. Documentation Priority: Due to the necessity of establishing the project's requirements, design, and implementation before actually beginning the actual work on the product, the documentation will take priority over the product. The only exceptions to this rule are the STD and the User Manual. The former will be developed after the SDD, meaning that it will overlap with product development. The latter will be developed at the end of the product development process, once the product is mostly finished, and will overlap both the end of the product development and the testing. (See Schedule for more details on how the various components will overlap.)

C. Communication and Meetings

Group members may communicate with each other via e-mail, telephone conversations, text messages, instant messenging, and physical meetings, both official and unofficial. Group members will exchange telephone contact information during the first official meeting of the project group, and may provide additional contact vectors as needed.

Official meetings will be decided upon at least one day in advance, and must last for at least one hour. The ideal period of time for meetings will be on Tuesdays or Wednesdays from 4:00 PM to 6:30 PM, but meetings may be scheduled for any point in time that does not conflict with University classes or other jobs' work schedules. Unofficial meetings follow the same pattern, but need not be constrained to the minimum time requirement that official meetings are.

Meetings with the Senior Project Professor and the Client will all be official. Meetings with the former will be held weekly in Room 2 of the EPL, Room L-309, or Room L-310 (Save during the weeks between terms or during university vacation time.) and will be logged via Meeting Notes (specs/Notas de Reuniones/Reuniones con Prof. Ortiz/) that will be signed by the Senior Project Professor, describing what has been done during the week and what should be done for the following week. Meetings with the latter will not be recorded in the same way; instead, the project group will meet with him with a number of questions and will end the meeting when said questions are answered and when all the requests by the client have been answered satisfactorily.

D. Schedule

Refer to Section 5.5 of this document for how the project's schedule will be organized.

E. Budget / Resources

Refer to Sections 5.3 and 5.4 for how the project's budget and resources will be used and allocated.

3.2. Assumptions, Dependencies, and Constraints

While creating this product, we have depended on a series of limitations that force our final result to be shaped the way it is. The following is a list of the assumptions and constraints we have

followed in the creation of this product and are based on the work mentioned in Section 2.4 of the SRS of this product.

Constraints:

1. The product must be able to process arbitrarily-sized data sets.
2. The product must not become unresponsive due to the processing of large data sets.
3. The product must be portable enough to be used on most platforms with minimal installation work.

Assumptions:

1. The programming language must follow with the same limitation as Constraint 1.
2. The system must accept files in the "CSV" format as user input.

Dependencies:

None. This project is not a branch of any other projects.

3.3. Risk Management

Problematic situations may occur throughout the duration of this project. This list will detail what are the measures that will be taken to minimize risk:

A. Project Members

1. Constant communication is mandatory, whether through phone, e-mail, or meetings on university grounds.

2. Whenever a meeting is scheduled, all project members must participate in that meeting. Persons that do not participate in at least half the group meetings without valid reasons will be removed from the project group.

3. Work will be divided evenly. In such a case that a member believes the task given to them are uneven or unsuited to them, they can bring up the complaint to the Project Leader via any communication medium.

4. If a project member does not participate in a meeting, they must be contacted via any communication medium to update them of the meeting's main topics, and the member must send confirmation of having received the information. This avoids leaving members without vital information.

5. All project members must perform roughly equal work to be credited properly. Any members that do not will be eliminated from the group. (For the purposes of this point, "roughly equal work" refers to a 50/50% split, with a variance of 15%.)

6. If the document-hosting service or the code-hosting service is terminated by either the Project Manager or a third party, the Project Manager's version of these documents or code will be used as the basis (Read: Master stream) of all future development. These files will then be uploaded onto another service with similar privacy protection as agreed upon by the Project Manager. Once this is done, this change must be communicated to all project members via any medium, and must explain the change and how to access the new versions.

B. Client

1. Meetings with the Client will occur when a milestone has been reached, but communication regarding the group's current progress should occur periodically so as to provide some amount of feedback to the Client.

2. Meetings with the Client will have their primary points written down so as to record the content of the discussions.

3. Features requested after the first meeting are not guaranteed to appear in the first version of the finished product. Instead, they will be accomodated as possible into the schedule.

4. If the Client wishes to drop the project, the project will continue to its conclusion, relying on the Senior Project Professor as backup client.

C. Senior Project Professor

1. Meetings with the Senior Project Professor will occur weekly on Thursdays at 4:00 PM to 4:30 PM either in Room 2 of the EPL lab, Room L-309's side room, or Room L-310. Project group members will come with a log of their weekly accomplishments and the following week's proposed tasks.

2. If a certain part of the project requires multiple rounds of feedback from the professor, additional meetings between the project member(s) and the professor should be scheduled throughout the week.

3. If the Senior Project Professor finds himself unable to continue work on the project or unwilling to provide adequate feedback on the components for multiple months for whatever reason, the project members will submit a request to the university (PUPR) for a new professor willing to provide assistance to the project.

3.4. Monitoring and Controlling Mechanisms

We will discuss how this product will be distributed and managed during its development in this section.

A. Project Repository

The PlasmaGraph project's files will all be distributed and managed via the project's GitHub page. Said service is a version control system designed to manage multiple versions of files, allowing for easy access while providing excellent error-correction mechanisms.

Due to the nature of the GitHub service, any person may make a copy of the project's contents and begin work on their own "fork" of the project. Therefore, simple access to the files, that is, being able to view the contents, of said repository will be available to anyone, but modification access to the contents of the main branch of the project will only be provided to the group members via the "Contributor" status. The Project Lead will only be able to grant this status, therefore, to people that request integration into the project group. Any group member that leaves the project will have his or her modification rights revoked. This way, access to the main branch of the project, called the "master" branch, will be protected from malicious users.

The project's GitHub web page is located in: https://github.com/CherimaeNemeta/PlasmaGraph

B. Repository Organization

The project's GitHub repository will be structured in the following format:

bin
docs
lib
specs
src
test

The "bin" directory will hold the compiled versions of the product. The "docs" directory will hold the JavaDocs created as a result of the code documentation. (See Work Package B1 in section 5.1 of this document for more details.) The "lib" directory will hold the additional libraries used by the project via the JAR files located within; linking to this via the Eclipse "External Libraries" feature will allow access to the support functions that they provide. The "specs" directory will hold the SRS / SDD / STD / SPMP / presentation documents, as well as their fragments and any support files relevant to the product's development or maintenance. This directory will also hold the meeting notes for the Client and the Senior Project Professor. The "src" directory will hold the entirety of the Eclipse workspace; this way, all the relevant files that directly form a part of the PlasmaGraph product will be easily saved in the repository. Finally, the "test" directory will hold the environment for testing the product, including sample test data and test template data.

C. Repository Management

Each project member will maintain their own personal version of the project's contents; this is known as a "branch" in GitHub. This will serve to isolate their changes and manage their individual work until it (their work) is ready to be added to the main branch's ("master") contents. This section will discuss how the project members will perform these changes, view their changes, and include their work into the project's "master" branch.

Note: Please refer to Git and GitHub documentation to learn how to perform some of these functions.

1. Documenting Changes (and format)

Once some amount of work is completed, a project member can use GitHub's (or Git's, in the case of Linux computers) "Commit" function to document said changes to that point.

Commits contain the full list of changes, allowing for easy viewing of the details changed per commit. However, the commit's message allows the member to write a summary of the changes made, facilitating future reviewing. The user must submit a descriptive message, noting the general changes that were made to the project. For example, if the user has programmed a prototype XML data cleaner for the product's code, then the member must write a commit message that mentions that they have created said prototype and describe (briefly) what it does.

Once at least one set of changes has been committed, the member may then move those changes over to their branch via the "Push" function. Since this project is not security-sensitive, either method for Pushing commits is acceptable. (That is, the HTTP or SSL URL links are both valid.)

2. Reviewing Changes

Changes can be reviewed via the GitHub project page's "Commits" viewing option. The commits of branches that aren't the main one can be accessed by navigating to the "Branches" viewing option and selecting the desired branch, then viewing the "Commits" for that branch.

3. Moving Changes to the Master Branch

When a sufficiently-large milestone has been reached in a branch (such as when a specification document like the SRS / SDD / STD / SPMP is completed), said data should be moved to the main branch via the Pull Request function. To do so, said project member must create a new Pull Request between the base branch, the "Master" branch, and the comparing branch, the project member's branch, and include the request's reason as a comment. Once done, the member will discuss the situation with the Project Lead, allowing the Project Lead to approve or deny it as she or he sees fit. If valid, the Project Lead will opt to merge the two. The project member will then have to create a new branch in

order to continue work on the project. This process allows other project members to routinely take advantage of new work related to the project without sacrificing much of the main branch's quality.

D. Problem Resolution

Solution to the various possible problems that may arise will be considered on a per-situation basis. Solutions such as large-scale deletion/merging, or drastic tools such as "rebase" will be used only when absolutely necessary. The Project Lead should discuss the options available with the Design Lead before deciding on a course of action.

3.5. Staffing Plan

A minimum of three persons will be required to complete this product; these persons, as well as the requirements for each, are as follows:

A. Project Lead

The Project Lead is the director of the group, orchestrating the meetings between client(s), professors, and group members, maintaining the SPMP, designing the testing to be done via the STD, and managing the tools used by the group.

This person must be a student at the university in which the Senior Project Professor works and must have the necessary course requirements to take the class "Computer Science Senior Project". Furthermore, the person must have previous experience with working with IEEE project design documents, namely the SRS, SDD, STD, and SPMP documents, and should have extensive university-level class experience in programming and program design. (Said experience should result from taking the required coursework for a Computer Science or Engineering Bachellor's degree.) Said person should also have some experience with managing a team, but this is not vital to the role of this member, as the group size is small.

B. Design Lead

The Design Lead is the other part of the success of this project; said person is the mastermind of the SRS, detailing how the product will interact with the external environment, and even working a lot with how it will perform its functions.

This person must be a student at the university in which the Senior Project Professor works and must have the necessary course requirements to take the class "Computer Science Senior Project". Furthermore, the person must have previous experience with working with IEEE project design documents, namely the SRS, SDD, STD, and SPMP documents, and should have extensive university-level class experience in programming and program design. (Said experience should result from taking the required coursework for a Computer Science or Engineering Bachellor's degree.)

C. Senior Project Professor

The Senior Project Professor is of equal importance to the success of the project as either the Project or Design Leads; s/he is the person that evaluates the project based on the IEEE documentation standards as well as coursework / official qualification standards. Without this person, the project cannot be labelled "completed" at any point in time.

This person must be a professor at the university in which the Project and Design Leads study, and must be officially recognized as a qualified person to evaluate the course "Computer Science Senior Project". Furthermore, the person must be well-acquainted with the project design standards (IEEE documentation; specifically, the SRS, SDD, STD, and SPMP document formats.), programming design standards (proper coding and documentation standards), and project management standards (group management, scheduling, and vital milestone requirements).

Any elimination of current personnel must be done with (1) a new qualified person ready for immediate assignment, and (2) the approval of all university faculty overseeing completion of the course, including the Senior Project Professor if s/he is not being substituted. Changes in personnel should be directed to the Client(s) in the interest of keeping them abreast of the group's status.

Training will be provided by the Project Lead or Design Lead to the new member(s) of the project group as needed.

4. Technical Process

4.1. Methods, Tools, and Techniques

The data-graping program PlasmaGraph must follow these requirements:

A. During Creation:

1. Operating System

The Operating System (OS) used to create this program must be any that can run the Eclipse IDE program.

2. Integrated Development Environment

The Integrated Development Environment (IDE) used to program and test this program will be the Eclipse IDE.

3. Programming Language

The Programming Language used to create this product will be the Java programming language; specifically, Java Version 7 will be used to create this program.

4. Code Repository

The code repository used for the creation of the final product, both documentation and program, will be the online version control system (VCS) GitHub. The service used from this website does not need to be any in particular; that is, either the public or private repository options may be used without worries.

5. Intermediary Document Format:

Each type of document will need to be in a specific format. The required formats are as follows:

a. Diagramming Document Format(s): .dia

b. Text Document Format(s): .txt, .doc, .docx, .md

c. Code Document Format(s): .java

d. Image Format(s): .png, .jpg

6. Document Repository

The document repository used for the creation of the final product, as mentioned above, will be GitHub. Note that the documentation repository will be located in the sub-directory named "specs".

B. During Usage:

1. Operating System

The Operating System (OS) used to run this program is any OS that contains a valid version of the Java Runtime Engine (JRE) or a similar Java Engine. For the purposes of this project, a valid version of Java refers to an updated version of Java 7 or above.

2. Java Version

The Java Version of any computer that wishes to run this program must be at least version 7. (JRE Version 7.) It is highly suggested to maintain an up-to-date version of the JRE on the computer running this program, however, so as to reduce potential Java-related problems.

4.2. Software Documentation

Documentation for this project will be split in three parts: the Software Documentation, the Test Documentation, and the Document Documentation. Each of these parts will be dedicated to describing or explaining certain features of either the product's capabilities, the product's design, or the project group's functioning with regards to creating the product. All these documents will be maintained via the GitHub version control system, as will every other document maintained.

a. Software Documentation

Software documentation is designed to explain what product code does in an efficient manner. Java provides the JavaDocs tool in order to ease the creation of such documentation, but it requires a specific format in order to generate said documentation. For the purposes of this project, all code must be documented as follows so as to standardize the generated documentation.

1. Where to Document: Each method must be documented via JavaDocs comment format. Otherwise, comments should be placed within methods to explain complex sections of code longer than 3 lines.

2. Basic Structure: JavaDocs comments start with the “/**” tag and end with the “*/” tag. Within the comment, there must be an explanation of the primary outputs, any exceptions that the method may throw, and any important notes regarding its implementation that may affect external entities. Afterwards, a number of tags must be used to properly document the inputs and outputs of the method.

Regular comments start with the “//” tag or are surrounded by the “/*” and “*/” tags. Their contents are not bound by any rules, so long as they explain what the following section of code does.

3. Tags to Use: The tag “@return” must always be used. Void functions will have a @return content of “Nothing.” The tag “@param” must be used if there is at least one input parameter for the method, and should specify exactly what format the input should be if there are any hidden nuances. The tag “@see” must be used if the method uses an important method of a different class, even if it is not a PlasmaGraph class. (For example, using a JfreeChart-related method.)

No specific tags are necessary for the interior of regular comments.

B. Test Documentation

Test documentation is designed to describe how and why certain tests are performed. Refer to the project's STD document for said documentation.

C. Project Documentation

Documents related to the product design and project group functioning count as Project Documentation.

The Version Control System used for this project, GitHub, will act as the changelog documentation for the purposes of documents, presentations, and product code. Said system provides the means to segregate changes made by individual project group members and revert changes on a per-document basis if needed.

Project Documentation must follow the IEEE guidelines as specified by the IEEE document manuals.

4.3. Project Support Functions

This project requires various minor functions in order to operate certain sections and guarantee quality.

A. Software Quality Assurance:

The quality of the final product will be maintained via three primary methods:

1. Mutual Oversight: As both the Project and Design Leads will be working on the code, they must both communicate to each other while they collaborate on the project, explaining how their contributions help progress to the final goal.

2. Software Documentation: All programmers (Project Lead and Design Lead) must document their code; this not only allows others to understand the product's internals better, but also allows everyone participating, themselves included, to catch the code's errors.

3. Testing: The product will be tested with various methods, both Black and White Box tests, and will be tested with both old data (data that has been provided by the client and used in tailoring the program) and new data (data that has been provided by the client, but not used in tailoring the program), creating both straightforward and blind tests that will test the robustness of the product.

B. Configuration Management:

Configuration of the systems, both while creating the product and in setting up the product's environment, is vital to the project; thus, there will be ample documentation on how to prepare the desired target environment for usage of the tools used and created by this project.

1. Initial IDE Configuration: Initial configuration of the development environment will be managed by a file that will explain how to properly prepare the environment. (Specifically, the "README.md" file that is present in the start of the GitHub page for the project contains details on how to set up the Eclipse IDE for proper usage.)

2. Final Product Configuration: Configuration details of the final product's environment will be detailed in the supplementary User Manual (Work Package B2) that will be provided with the product.

5. Work Packages, Schedule, and Budget

5.1. Work Packages

As mentioned in this project's SRS, the requirements for PlasmaGraph stipulate a number of important functions that it must perform. This product must include the following modules:

A. PlasmaGraph: The product. This package contains a number of important sub-packages.

1. Plasma Data Sanitizer: The product must take as input a CSV file, read and analyze its contents, edit (or purge) the data as necessary in order to standardize it, and the provide necessary vector to pass said data to the next module, the Data Graph Interface Module.

This module consists of a Data Sanitizer sub-module and the transport code sub-module that manages data through the product.

2. Data Graph Interface Module: This module must take as input the data provided to it by the Plasma Data Sanitizer. With it, this module will obtain from the user data regarding how s/he wishes to graph the sanitized data, and graph it in said manner. It must also provide a GUI for editing said options, and the ability to save both the options chosen and/or the graph as an image.

This module consists of the program's GUI sub-module and back-end code, and the Graph sub-module. The GUI sub-module will include the image-saving and template-loading/saving functions.

B. Product Documentation: Documents that will explain how to use the product correctly, and how to resolve potential problems.

1. JavaDocs: Internal product documentation that will make editing the product in the future easier, in the case of errors in need of fixing or additional features that will be included. These are generated from in-code comments, and only require minor work in order to create.

2. User Manual: External product documentation that will describe why the product exists, what it is, and how to use it.

C. Software Specification Documents: The supporting documentation for the product, specifying the design of the product and the group responsible for completing it.

1. Software Requirements Specification (SRS): Describes the product from an external viewpoint. It includes what it does and how it does it, and elaborates on requirements the product needs to be used.

2. Software Design Description (SDD): Details the product's internal structure and processes.

3. Software Test Document (STD): Details all testing the product undergoes, and the degree of success or failure on each.

4. Software Project Management Plan (SPMP): Details the project group's structure, composition, and behavior. Explains risks inherent to the project's completion, and steps taken to reduce said risk.

5. Product / Project Description Presentation: Showcases the work performed in the SRS and SPMP documents. Describes what the product is, what it will do, and how the project group will work in order to complete said product. This item will be used in the mid-schedule presentation of this project, before actual work on the product begins.

6. Final Project Presentation: Showcases the work performed in completion of the product. Performs similar work to that of the "Product / Project Description Presentation", but includes information regarding internal workings, test completion, the complete schedule of the project, and an explanation of how the product works. This item will be used in the final presentation of this project.

The component parts for the Software Specification Documents (C) are detailed in the IEEE 830 / 1016 / 829 / 1058 documents provided by the Senior Project Professor.

All presentations will be created via a "Powerpoint" program and will contain information from other work packages, carefully selected so as to showcase important aspects of the project's progress.

5.2. Dependencies

The following list details the dependencies for each of the work packages specified in Section 5.1 of this document:

Note: All dependency lists for each sub-package are written in order of appearance on this list.

A. PlasmaGraph

1. Plasma Data Sanitizer: SRS (C1), SDD (C2), SPMP (C4)
2. Data Graph Interface Module: SRS (C1), SDD (C2), SPMP (C4)

B. Product Documentation

1. JavaDocs: SRS (C1), SDD (C2), SPMP (C4)
2. User Manual: PlasmaGraph (A), JavaDocs (B1), SRS (C1), SDD (C2), STD (C3), SPMP (C4)

C. Software Specification Documents

1. Software Requirements Specification (SRS): None
2. Software Design Description (SDD): SRS (C1), SPMP (C4)
3. Software Test Document (STD): PlasmaGraph (A), SRS (C1), SDD (C2), SPMP (C4)
4. Software Project Management Plan (SPMP): None
5. Product / Project Description Presentation: SRS (C1), SPMP (C4)
6. Final Project Presentation: PlasmaGraph (A), Product Documentation (B), SRS (C1), SDD (C2), STD (C3), SPMP (C4), Product / Project Description Presentation (C5)

Please refer to Appendix A: Work Project Tree for more details.

Further details on how the packages will be developed can be found in "Appendix B: Project Gantt Chart".

5.3. Resource Requirements

The following resources will be needed for the completion of this project:

A. Hardware and Software

1. Hardware

The project group will utilize their own computers (laptops and/or desktops) to complete all work packages as detailed in section 5.1 of this document.

2. Software

The project group will use a number of software programs in order to complete the work packages.

- Dia / LibreOffice Draw: Diagramming software.
- Microsoft Word / LibreOffice Writer / SublimeText 2: Document-writing software.
- Microsoft PowerPoint / LibreOffice Impress: Presentation-making software.
- LibreOffice Calc: Table-making software.
- SublimeText 2 / Eclipse / NetBeans IDE: Programming software.
- Git (via GitHub): File Version Control Software service.
- GIMP / Paint.NET / MSPaint: Image manipulation software.
- Mozilla Firefox / Google Chrome: Internet browsing software.

B. Temporary and Permananent Locations

1. Temporary Locations

The following locations will be used:

- PUPR Campus - Library - Study Rooms: Project group meetings.
- PUPR Campus - Classrooms L-310, EPL-2: Meetings with Senior Project Professor.
- PUPR Campus - Plasma Laboratory: Meetings with Client.
- PUPR Campus - <final presentation room>: Final Presentation.

2. Permanent Locations

No permanent locations will be needed for the purposes of finishing this project.

C. Human Resources

The Project and Design Leads will work on all work packages together, save the SRS (Design Lead) and SPMP. (Project Lead) The Project Lead will manage the GitHub file repository. Presentations will be given by both members.

The Senior Project Professor will be used for the entirety of the project as a reference to guide the project correctly.

D. Meeting Logs

Meeting Logs with the Senior Project Professor and Client will be maintained by the Project Lead and will be provided in the final package along with the rest of the items.

E. Time

The time necessary for the completion of this project, as well as for each individual Work Package, is detailed in Appendix A: Project Gantt Chart.

5.4. Budget and Resource Allocation

The resources required (and their budget allocated) are as described in Appendix B: Project Budget.

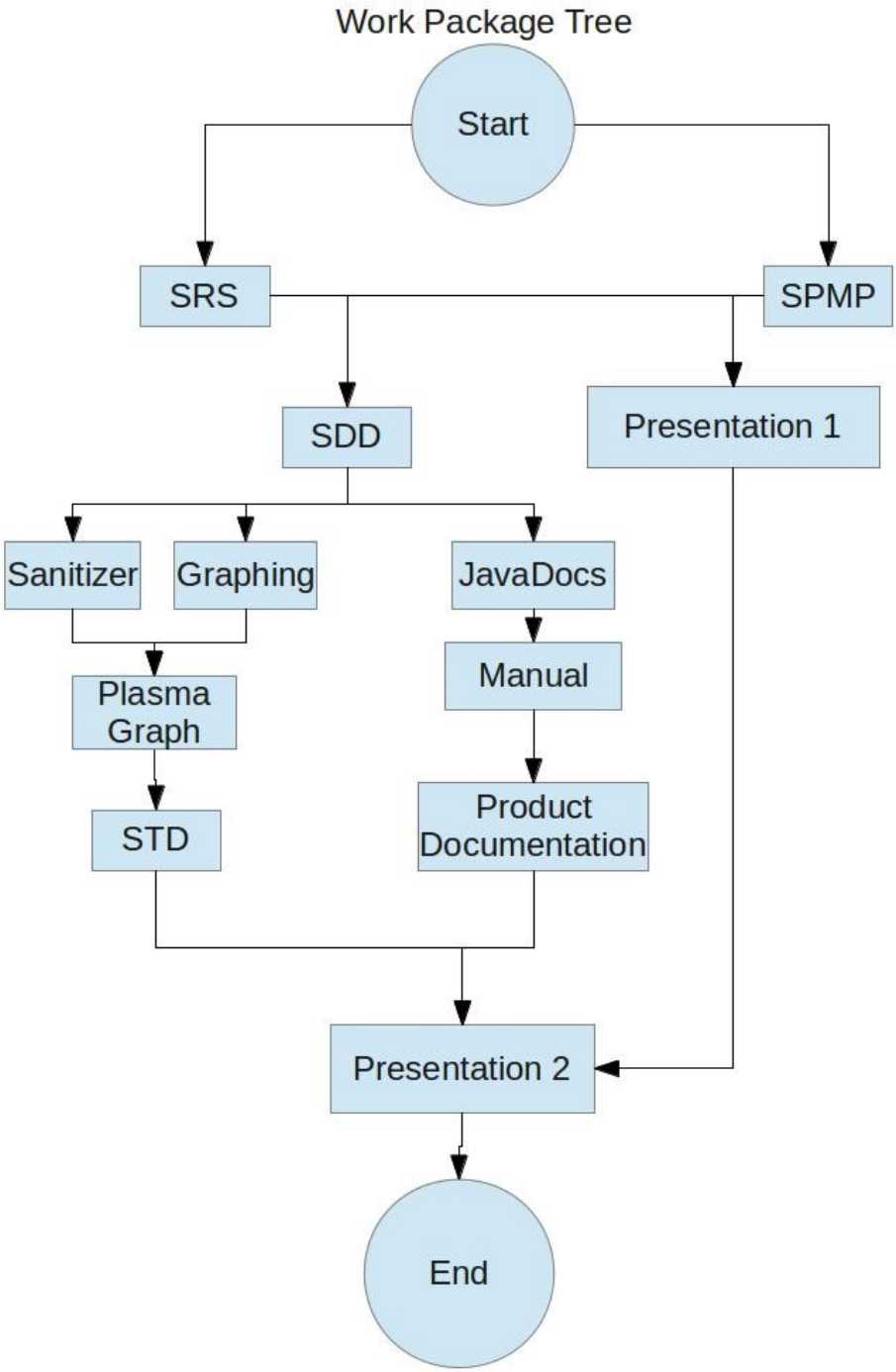
5.5. Schedule

The schedule that the project will adhere to is described in Appendix A: Project Gantt Chart.

Appendices

Appendix A: Work Package Tree

The following graph describes the tree-like structure of how work packages will be developed.



Notes:

"Plasma Data Sanitizer" is labeled "Sanitizer".

"Data Graph Interface Module" is labeled "Graphing".

"User Manual" is labeled "Manual".

"Product / Project Description Presentation" is labeled "Presentation 1".

"Final Project Presentation" is labeled "Presentation 2".

Appendix B: Project Gantt Chart

<Insert “gantt_chart.jpg”.>

Appendix C: Project Budget

<Insert “budget.jpg”.>

Index