# **System Requirements Specification**

## **Dependency Graph - CS490 - Fall 2020**

***Team Members:***

* Aaron Van De Brook
* Corrina Del Greco
* Luis Mora
* Samantha Shultz

|  |  |
| --- | --- |
| Version | Date |
| 1.0 | 10.1.20 |
| 2.0 | 10.29.20 |
| 3.0 | 11.30.20 |

***Contents of this Document***

[Introduction](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#section-1-introduction)

[System to be Produced](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#system-to-be-produced)

[Definitions, Acronyms, and Abbreviations](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#definitions-acronyms-and-abbreviations)

[Product Overview](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#section-2-product-overview)

[Assumptions](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#assumptions)

[Stakeholders](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#stakeholders)

[Use Case Diagram](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#use-case-diagram)

[Use Case Descriptions](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#use-case-descriptions)

[Specific Requirements](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#section-3-specific-requirements)

[Functional Requirements](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#31-functional-requirements)

[Interface Requirements](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#32-interface-requirements)

[Physical Environment Requirements](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#33-physical-environment-requirements)

[User and Human Factors Requirements](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#34-user-and-human-factors-requirements)

[Documentation Requirements](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#35-documentation-requirements)

[Data Requirements](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#36-data-requirements)

[Resource Requirements](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#37-resource-requirements)

[Security Requirements](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#38-security-requirements)

[Quality Assurance Requirements](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#39-quality-assurance-requirements)

[Supporting Material](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#section-4-supporting-material)

### 

### 

### 

### 

### 

### 

### **Section 1: Introduction**

#### **System to be Produced**

The system has the capability to create visual representations of the dependencies between modules, to assist in reducing the amount of work necessary to refactor modules. The system shall generate dependency graphs based on Graphviz dot file(s). Doxygen can generate call graphs of a project, but these call graphs can be quite large depending on the project. Our system will take the text representations of these call graphs and construct simplified dependency graphs. The simplified graphs will include the module for which the functions belong to, and they will separate functions between public and private.

### **Definitions, Acronyms, and Abbreviations**

* Doxygen - A tool for generating software reference documentation, used to generate Dot file(s) from a project for the system to use
* Dot - A software language to describe directed graphs
* Javadoc - A documentation generator for generating API documentation in HTML format from Java source code
* JDK - Java Development Kit

### **Section 2: Product Overview**

#### **Assumptions**

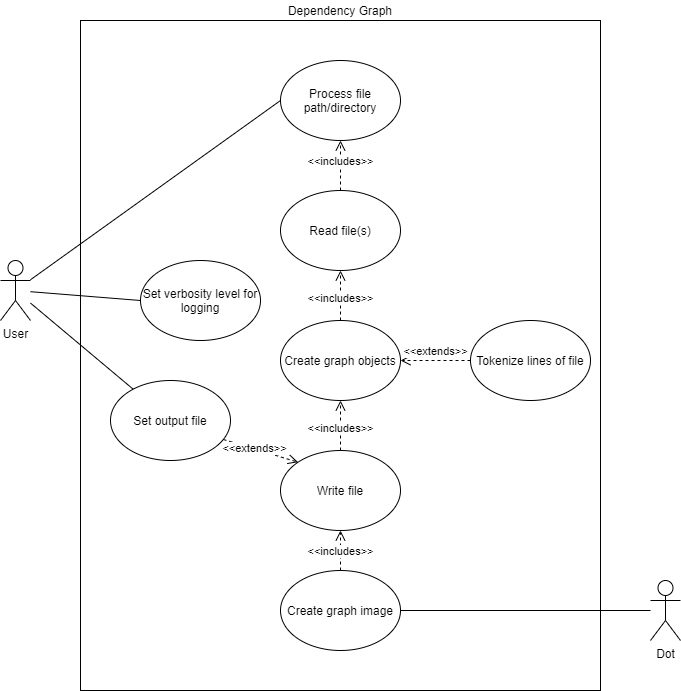
* The user has used Doxygen to generate Dot file(s) for the project that they wish to see graphs for
* The functions in the project should have a naming convention as follows: MOD\_\* for public functions and mod\_\* for private functions where mod is the prefix for the module the function belongs to, and \* is a placeholder for the function name.
* The user will run on on Windows

#### **Stakeholders**

Jianhua Liu: Customer plans to use the system to assist in cleaning up personal research code before making it open source.

#### 

#### **Use Case Diagram**

****

**Figure 1: Use Case Diagram**

#### **Use Case Descriptions**

Notably, our program does not have a variety of actors, only the general user provides a value through the command line to the program. This command line argument provides either a single Dot file or a directory containing Dot files. Internally, these are manipulated into something that Dot may create a desirable output image of. At a high level, program sequence can be viewed sequentially (predictably from one class to the next). All of the classes mentioned below (Reader, Parser, Lexer, Writer) are explained in detail in the System Design Document.

* Use Case 1: The user gives a file path to the program through the command line and it is processed.
* Use Case 2: The Reader reads the Dot file(s) found at the path and provides the file contents to Parser.
* Use Case 3: The Parser creates graph objects from the file contents, and removes and adds connections as needed.
* Use Case 4: The Lexer assists the Parser by tokenizing lines of the file.
* Use Case 5: The Writer uses the final graph objects to write new Dot file(s).
* Use Case 6: Dot outputs a visual image as output using the new Dot file(s).
* Use Case 7: The user sets the verbosity level of the program's logging output.
* Use Case 8: The user sets the name/destination for the output image file.

### **Section 3: Specific Requirements**

#### **3.1 Functional Requirements**

|  |  |
| --- | --- |
| **No.** | **FR1** |
| Statement | The user shall provide a single Dot file for the system to create a graph image for |
| Source | The Customer |
| Dependency | None |
| Conflicts | None |
| Supporting Material | Use Case Diagram |
| Evaluation Method | Command line, -s flag |
| Revision History | Created version 3.0 |
| Where to Implement | Configurator |

|  |  |
| --- | --- |
| **No.** | **FR2** |
| Statement | The user shall provide a directory containing Dot file(s) for the system to create a single graph image for |
| Source | The Customer |
| Dependency | None |
| Conflicts | None |
| Supporting Material | Use Case Diagram |
| Evaluation Method | Command line, -d flag |
| Revision History | Created version 3.0 |
| Where to Implement | Configurator |

|  |  |
| --- | --- |
| **No.** | **FR3** |
| Statement | The user shall be able to set the logging verbosity level |
| Source | The Development Team |
| Dependency | Java Util Logging package |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Command line, -v flag |
| Revision History | Created version 3.0 |
| Where to Implement | Configurator |

|  |  |
| --- | --- |
| **No.** | **FR4** |
| Statement | The user shall be able to set the output log file |
| Source | The Development Team |
| Dependency | Java Util Logging package |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Command line, -L flag |
| Revision History | Created version 3.0 |
| Where to Implement | Configurator |

|  |  |
| --- | --- |
| **No.** | **FR5** |
| Statement | The user shall be able to set the output graph file |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Command line, -o flag |
| Revision History | Created version 3.0 |
| Where to Implement | Configurator |

|  |  |
| --- | --- |
| **No.** | **FR6** |
| Statement | The program shall group functions by public or private |
| Source | The Customer |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | View output graph |
| Revision History | Created version 1.0 |
| Where to Implement | Parser |

|  |  |
| --- | --- |
| **No.** | **FR7** |
| Statement | The program shall not show dependencies between functions within a module |
| Source | The Customer |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | View output graph |
| Revision History | Created version 3.0 |
| Where to Implement | Parser |

|  |  |
| --- | --- |
| **No.** | **FR8** |
| Statement | The program shall not have global variables |
| Source | The Customer |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Check for use of getter/setter functions in the code |
| Revision History | Created version 1.0, revised version 3.0 |
| Where to Implement | All classes |

|  |  |
| --- | --- |
| **No.** | **FR9** |
| Statement | The program shall output a dependency graph image |
| Source | The Customer |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | View output graph |
| Revision History | Created version 3.0 |
| Where to Implement | All classes |

#### **3.2 Interface Requirements**

|  |  |
| --- | --- |
| **No.** | **IR1** |
| Statement | The user shall use the program in its entirety through the command line interface |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | Use Case Diagram |
| Evaluation Method | Run the program |
| Revision History | Created version 1.0 |
| Where to Implement | Configurator |

|  |  |
| --- | --- |
| **No.** | **IR2** |
| Statement | The program shall organize the functions in a module in a neat way |
| Source | The Customer |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | View output graph |
| Revision History | Created version 1.0 |
| Where to Implement | Writer |

#### **3.3 Physical Environment Requirements**

|  |  |
| --- | --- |
| **No.** | **PER1** |
| Statement | The user shall run the program on Windows |
| Source | The Development Team |
| Dependency | All |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Run the program |
| Revision History | Created version 1.0 |
| Where to Implement | All Classes |

|  |  |
| --- | --- |
| **No.** | **PER2** |
| Statement | The user shall have a JDK of 14 or higher |
| Source | The Development Team |
| Dependency | All except PE1 |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Check the Java version |
| Revision History | Created version 1.0, revised version 3.0 |
| Where to Implement | All Classes |

#### **3.4 User and Human Factors Requirements**

|  |  |
| --- | --- |
| **No.** | **UHR1** |
| Statement | The user shall be able to access a help menu |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Command line, -h argument |
| Revision History | Created version 1.0, revised version 3.0 |
| Where to Implement | Configurator |

|  |  |
| --- | --- |
| **No.** | **UHR2** |
| Statement | The program shall inform the user of a wrongly inputted file |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | Use Case Diagram |
| Evaluation Method | Command line, -s flag, file with extension that is not .dot or non-existent Dot file |
| Revision History | Created version 1.0, evaluation method revised in version 3.0 |
| Where to Implement | Configurator |

|  |  |
| --- | --- |
| **No.** | **UHR3** |
| Statement | The program shall inform the user of a wrongly inputted directory |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | Use Case Diagram |
| Evaluation Method | Command line, -d flag, non-existent directory |
| Revision History | Created version 3.0 |
| Where to Implement | Configurator |

#### **3.5 Documentation Requirements**

|  |  |
| --- | --- |
| **No.** | **DoR1** |
| Statement | The programmers shall document the code using Javadoc |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | View generated documentation |
| Revision History | Created version 1.0 |
| Where to Implement | All Classes |

#### **3.6 Data Requirements**

|  |  |
| --- | --- |
| **No.** | **DR1** |
| Statement | The programmers shall store graph attributes in objects |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Check the code |
| Revision History | Created version 3.0 |
| Where to Implement | Parser |

|  |  |
| --- | --- |
| **No.** | **DR2** |
| Statement | The programmers shall have a knowledge of tree data structures |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Check for correct syntax in Dot files |
| Revision History | Created version 2.0 |
| Where to Implement | All Classes |

#### **3.7 Resource Requirements**

|  |  |
| --- | --- |
| **No.** | **RR1** |
| Statement | The programmers shall package the end project into a Java archive (JAR) |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Run the program without downloading/installing any external programs |
| Revision History | Created version 1.0 |
| Where to Implement | All Classes |

#### **3.8 Security Requirements**

|  |  |
| --- | --- |
| **No.** | **SR1** |
| Statement | The program shall log behavior to a file |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | Use Case Diagram |
| Evaluation Method | Run the program, check output log file |
| Revision History | Created version 3.0 |
| Where to Implement | All Classes |

|  |  |
| --- | --- |
| **No.** | **SR2** |
| Statement | The programmers shall close all open file buffers when they are not using them |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Check the code |
| Revision History | Created version 1.0 |
| Where to Implement | Reader |

#### **3.9 Quality Assurance Requirements**

|  |  |
| --- | --- |
| **No.** | **QA1** |
| Statement | The programmers shall not hardcode file paths |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Run the program on several different computers |
| Revision History | Created version 1.0 |
| Where to Implement | All Classes |

### **Section 4: Supporting Material**

* [Abstract Program Flowchart](https://github.com/AVanDeBrook/c-dependency-graph/wiki/Abstract-Program-Flowchart)
  + Lays out the basic structure of the main method
* [Dot Language Notes](https://github.com/AVanDeBrook/c-dependency-graph/wiki/The-DOT-Language-Notes)
  + Formal language definition of dot
  + Influences the design of the parser and writer, both need to conform to Dot's language specification
* [Parser Design](https://github.com/AVanDeBrook/c-dependency-graph/wiki/Parser-Design)
  + Lays out the initial basic design for the parser
  + The parsers design will have an influence on the design of other classes such as Reader and Manipulator as well as the Manager
  + Parser is complex and easy to mess up, so having a detailed and thorough design will prevent future issues