# **System Requirements Specification**

### **Dependency Graph**

### **CS490, Fall 2020**

*Team Name:* The Configurators

*Team Members:*

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

*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)
  + [Applicable Standards](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#applicable-standards)
* [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)
  + [Event Table](https://github.com/AVanDeBrook/c-dependency-graph/wiki/SRS-Document#event-table)
  + [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**

* A system that has the capability to create visual representations of the dependencies between modules to reduce the amount of work necessary to refactor modules. The system shall generate dependency graphs based on Graphviz dot file(s).

#### **Applicable Standards**

* Open JDK must be 14 or later
* Must run with Windows

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

* JDK - Java Development Kit
* Doxygen - a documentation generator, a tool for writing software reference documentation
* Javadoc - a documentation generator for generating API documentation in HTML format from Java source code
* Hardcode - fix (data or parameters) in a program in such a way that they cannot be altered without modifying the program

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

#### **Assumptions**

* The user has generated the Doxygen dot file(s)
* The user will run with Windows

#### **Stakeholders**

* Jianhua Liu (Customer) - Wants to clean up personal research code, and plans to use this program to help with that
* Ilhan Akbas (Senior Management) - Comes from a professor standpoint, wants success of group
* Shafagh Jafer (Senior Management) - Comes from a professor standpoint, wants success of group

#### **Event Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Event Name** | **External Stimuli** | **External Responses** | **Internal Data and State** |
| Gives a Single File | Command line argument | Produce dependency graph | Single file passes to the reader class |
| Gives a Folder of Files | Command line argument | Produce dependency graph | Name of directory passes to the reader class |
| Gives Wrong File(s) | Command line argument | Inform user of wrong file input | Reader class will return back to Manager class |

#### 

#### 

#### 

#### 

#### 

#### 

#### 

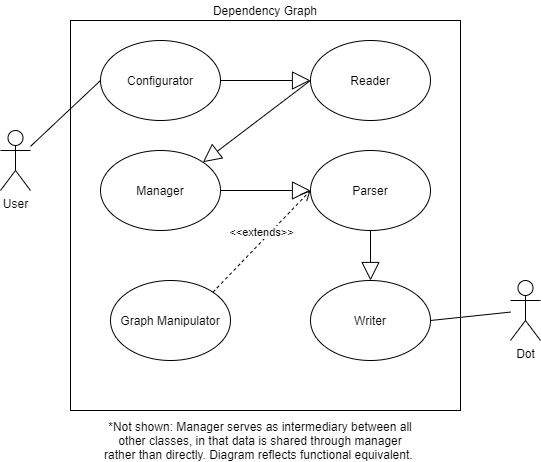
#### 

#### 

#### 

#### 

#### **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), but the Manager will serve as an intermediary between classes so that they do not pass data directly between the two. For readability purposes, the Use Case Diagram does not include this detail. Instead, the manager is only showed where plays a role in transforming the data for the next class.
* Use Case: User gives single file argument to program through command line. Configurator accepts this.
* Use Case: User gives directory argument to program through command line. Configurator accepts this.
* Use Case: Configurator gives a file path to the Reader.
* Use Case: Reader reads the dot file(s) at the file path and provides the file contents to the Manager.
* Use Case: The Manager feeds the file contents one by one to the Parser.
* Use Case: The Parser creates graph objects from the file contents, the Graph Manipulator alters these objects to remove and add connections as needed.
* Use Case: The final graph objects are provided to the Writer.
* Use Case: The Writer writes a file which Dot can create a visual image of as output.
* See the [Program Flow Design](https://github.com/AVanDeBrook/c-dependency-graph/wiki/Program-Flow-Design) wiki for more.

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

#### **3.1 Functional Requirements**

|  |  |
| --- | --- |
| **No.** | **FR1** |
| Statement | The program shall return a statement informing the user of wrongly inputted file(s) |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | Use Case Diagram |
| Evaluation Method | By putting in a non-dot file |
| Revision History | TBD |

|  |  |
| --- | --- |
| **No.** | **FR2** |
| Statement | The program shall have a way of checking if a public function is really public |
| Source | The Customer |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | By running written program to search for public functions in a module |
| Revision History | TBD |

|  |  |
| --- | --- |
| **No.** | **FR3** |
| Statement | The program shall remove all global variables |
| Source | The Costumer |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | By checking for the getter/setter functions |
| Revision History | TBD |

#### **3.2 Interface Requirements**

|  |  |
| --- | --- |
| **No.** | **IR1** |
| Statement | The user shall be able to 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 | By running the program |
| Revision History | TBD |

|  |  |
| --- | --- |
| **No.** | **IR2** |
| Statement | The program shall organize the functions listed in a module in a neat way |
| Source | The Customer |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | By looking at printed modules and analyzing display |
| Revision History | TBD |

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

|  |  |
| --- | --- |
| **No.** | **PER1** |
| Statement | The user shall run the program with Windows |
| Source | The Development Team |
| Dependency | All |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | By running the program |
| Revision History | TBD |

|  |  |
| --- | --- |
| **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 | By running the Java version |
| Revision History | TBD |

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

|  |  |
| --- | --- |
| **No.** | **UHR1** |
| Statement | The user shall be able to access the “help” menu through the command line |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Run the argument in the command line |
| Revision History | TBD |

#### **3.5 Documentation Requirements**

|  |  |
| --- | --- |
| **No.** | **DoR1** |
| Statement | The programmers shall write a Javadoc |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | View generated documentation |
| Revision History | TBD |

#### **3.6 Data Requirements**

|  |  |
| --- | --- |
| **No.** | **DR1** |
| Statement | The programmers shall have a general knowledge of tree data structures |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Syntactically correct dot files |
| Revision History | TBD |

#### **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 | Running the program without having to download/install any external programs |
| Revision History | TBD |

#### **3.8 Security Requirements**

|  |  |
| --- | --- |
| **No.** | **SR1** |
| Statement | The program shall log file accesses |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | Use Case Diagram |
| Evaluation Method | Run program, read standard output |
| Revision History | TBD |

|  |  |
| --- | --- |
| **No.** | **SR2** |
| Statement | The Development Team 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 | TBD |

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

|  |  |
| --- | --- |
| **No.** | **QA1** |
| Statement | The system shall detect and isolate incorrect file(s) and/or directory/directories |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Run program with faulty file(s) |
| Revision History | TBD |

|  |  |
| --- | --- |
| **No.** | **QA2** |
| Statement | The programmers shall not hardcode file paths |
| Source | The Development Team |
| Dependency | None |
| Conflicts | None |
| Supporting Material | None |
| Evaluation Method | Run program on several different computers |
| Revision History | TBD |

### **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
  + Also, a good jumping off point for creating classes
* [Dot Language Notes](https://github.com/AVanDeBrook/c-dependency-graph/wiki/The-DOT-Language-Notes)
  + Important for designing the parser and creating an example format for the graphs
* [Parser Design](https://github.com/AVanDeBrook/c-dependency-graph/wiki/Parser-Design)
  + Lays out at least a 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 screw up, so having a detailed and thorough design will prevent future pains
* [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