VMWare Virtual Network Visualizer

Software Requirements Specification

Version 4.0

November 17, 2012

Team
–Intentionally Left Blank–

Pierce Trey, Everett Bloch,
Jonathan Lamb, Daniel Palmersheim

Prepared For
UI CS 383—Software Engineering I
Instructor: Paul Oman, Ph.D.
Fall 2012

SRS template acquired from www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-**srs**-v1.doc

Revision History

Date	Description	Author	Comments
9/27/12	1.0	ILB	Initial submition
10/30/12	2.0	ILB	Updates and revisions
11/11/12	3.0	ILB	Updates and revisions
11/17/12	4.0	ILB	Completion of Document

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Dr. Alves-Foss	Customer Contact	

Table of Contents

1. Introduction

The VMware Virtual Network Visualizer provides a means to view an interconnected network of computers in an interactive, easy to use GUI. It supplies a wide array of information pertaining to the network and the individual machines operating on the network.

1.1 Purpose

Having a visual representation of an operating network allows the user to account for all of the machines using the network, and provides information about these machines and their connectivity for maintenance and management purposes.

1.2 Scope

The VMware Virtual Network Visualizer displays a predefined network of nodes. Its functionality is to obtain information about each machine on a network through an input file that specifies each machine operating under a unique IP address, and display it as a node in the GUI. Neighboring nodes will be indicated by an edge connecting them, generating a web of interconnected nodes that simulate the workings of the network. The user will have the ability to manipulate the visual output of the graph by moving nodes, rotating the graph, and skewing the graph for a better visualization of the data.

This product's intended use is to be a visualization of the operating network, it will not have any functionality to add or drop machines from the network or fix any problems related to bottlenecks, latency, or other issues that it uncovers.

Having this information on hand is useful for management and maintenance of the operating network; providing the number of machines using the network, their respective weights and capacities between other machines, and the general structure of the network to the user. With this information the user will have knowledge of potential bottlenecks in the network or problems with a specific machine on the network so to keep it operating efficiently.

1.3 Definitions, Acronyms, and Abbreviations

ILB – Intentionally Left Blank

GUI – Graphical User Interface

JDK – Java Developers Kit

JRE – Java Run-time Environment

JUNG – Java Universal Network/Graph library.

XML – Extensible Markup Language

1.4 References

IEEE 830 Recommended Practice for Software Requirements

Software Design Document (SDD) - Intentionally Left Blank

1.5 Overview

The rest of this document goes into further detail about the general description of the product, its specific requirements, the analysis models for those requirements, and the procedure for changing this document. The general descriptions give an overview of the functionality and characteristics of the product for the user. The specific requirements are more in depth descriptions of the requirements of the product.

2. General Description

2.1 Product Perspective

This product is similar to Nmap and its derivatives in the since that they all identify a selected network and provide information about the machines on that network. The difference is that this is a more refined product that provides basic information about a network in a intuitive GUI, that allows for analytical algorithms, such as shortest weighted and unweighted paths, used to discover the number of hops or bandwidth between two machines. Node information may be displayed through a node info query feature that displays the nodes IP along neighboring node IPs and connectivity information.

2.2 Product Functions

The graph will display a node for each machine and an edge connecting two connected machines on the network.
User functions for the VMware Virtual Network Visualizer are provided through the GUI. Information about each node will be produced when the ID is queried in the get node info input box. Shortest weighted and unweighted path dialog boxes will be displayable for network analysis. Furthermore, the graph will be able to be saved as a PDF for later review.

2.3 User Characteristics

This product is geared toward people with a knowledge of networking environments.

2.4 General Constraints

Input file structure:
Input files need to be consistent, according to specifications given to the user. It is assumed that the input will be consistent for every node.

2.5 Assumptions and Dependencies

This product was created using JDK 7, an assumption is that the user has the latest version of the JRE installed on their computer.
Network conditions must match input file. Input files are XML files generated from an Nmap scan specifying IP addresses and neighboring IP addresses. The user is expected to

generate this XML file prior to execution, any errors will be displayed in the session log and log_file.txt. An example XML file will be provided as a default file.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface will be a graph created using the JUNG library, along with buttons and dialog buttons to assist the user in the functionality of the product.

3.1.2 Software Interfaces

The software will interface with a specific XML file format for input, and will output this information through the GUI, via the graph and dialog boxes.

3.2 Functional Requirements

3.2.1 Input File

The input file will be an XML file converted from an Nmap scan of all of the machines on the network. This file will contain the IP address of each machine, along with the neighboring machines to specify who is connected to who. For each neighboring node, there will be a weight and capacity included for the connection between the two machines. This XML file will contain all of the input information needed to generate the graph for the underlying network. A default file will be provided as a format example.

3.2.2 Input Processing

The input file will be in XML format with fields specifying each data area that will be used to create the graph. The process for bringing in the input will be to parse the XML file into a Data Payload class that stores each machine's info in a separate object. Each object will be stored in a dynamic data structure to be used in graph generation.

3.2.3 Graph Generation

Once the Data Payload data structure has been created, the network data can be used to generate the graph using JUNG's Graph class creating data fields for vertices and edges. In order to prevent duplicate edges between vertices, each input Data Payload object will be recorded, keeping track of which vertex will have what neighbors. Where each vertex will have an edge between each of its neighbors. If there is ever a duplicate vertex to neighbor combination the most recent edge will be ignored. This process will create the graph data for the JUNG visualizer.

3.2.4 Graph Visualizer

Once the Graph data has been created, a visualizer can be crated for it. The Visualization Viewer class in JUNG uses the previously created Graph class to display the underlying data in a user friendly format. Each vertex is indicated

with a circle, and an edge a line between two vertices. The Visualization Viewer allows for the user to manipulate the graph by moving vertices around with the mouse, rotating the orientation of the graph, zooming in and out with the mouse wheel, skewing the view of the canvas, and dragging the graph around the canvas. These features will be provided via two separate mouse modes, Picking and Transforming. Picking allows for manipulation of the vertices and edges, where transforming allows for manipulation of the canvas. A button will allow the user to reset the graph to its original position. Along with these features, a name can be specified with the each vertex and edge which can be displayed in the visualizer as a label. A button will allow the user to toggle the vertex and edge labels off or on, to help reduce clutter in a large graph or to provide more information on the graph respectively.

3.2.5 Graph Algorithms

The Menu Bar at the top of the window frame will have an algorithm drop down box to provide algorithms to be performed on the graph. These algorithms will create a dialog box for the user to input the specified information. The Unweighted Shortest Path algorithm has the user supply two nodes and it returns the shortest path through the network from the first node to the second, providing the path with the least amount of hops. The Weighted Shortest Path algorithm has the user supply two nodes and returns the shortest path between the two nodes containing the smallest weight. This provides a path with the quickest hop time between two machines, ie. smallest weight.

3.2.6 Saving a JPG

The Menu Bar at the top of the window frame will have a Save as JPG feature, which when clicked will ask for a location and name of saved file to save the JPG to. This feature will save a JPG of the current state of the graph visualizer in the specified folder with the specified name.

3.2.7 Logging

All of these features will be logged upon creation in a log file. In addition a temporary session log window will be available for the user to observe the progress of the logged information in the current session. This window will act as a progress meter on start up, to show the progression of the program before the user interface and graph visualizer will be visible.

3.2.8 Help Menu

A help menu will be provided at the top of the window frame which will display a PDF of this product's User Manual, for a quick reference of features and functionality.

3.3 Use Cases

3.3.1 Network Surveying (realized in SDD 5.1)

One major use of this product is for network surveying. Using this product to look at the network and play around with the various features to get a better

understanding of the network and the way it is operating. Providing a very simple means to overview the underlying network.

3.3.2 Network Management (realized in SDD 5.2)

Another use of this product is for Network Management, using all of the reasons as stated for network surveying but geared more toward the efficient operation of the network. Identifying areas of improvement and potential problems in the underlying network.

3.4 Non-Functional Requirements

3.4.1 Performance

Initial start up of this product may vary depending on the size of network the user is attempting to generate. A session log, upon start up, will display information regarding the progression of the program so to indicate to the user that the program is being initialized. Similarly, the graph algorithm run time may vary depending on the size of the network the algorithm must analyze.

3.4.2 Reliability

This program is running on the heavily used JRE, so it should be reliable due to the stable nature of the platform. Java is also a very reliable language that takes care of many reliability issues with its excellent garbage collection and other automagic devices.

3.4.3 Availability

This program will be available as executable and source code on our team github site: https://github.com/celloman/ILB-Network-Visualizer

3.4.4 Maintainability

Upon completion of project and end of semester, the VMware Virtual Network Visualizer will be available at our team github repository. This product is subject to no further maintenance upon completion of project.

3.4.5 Portability

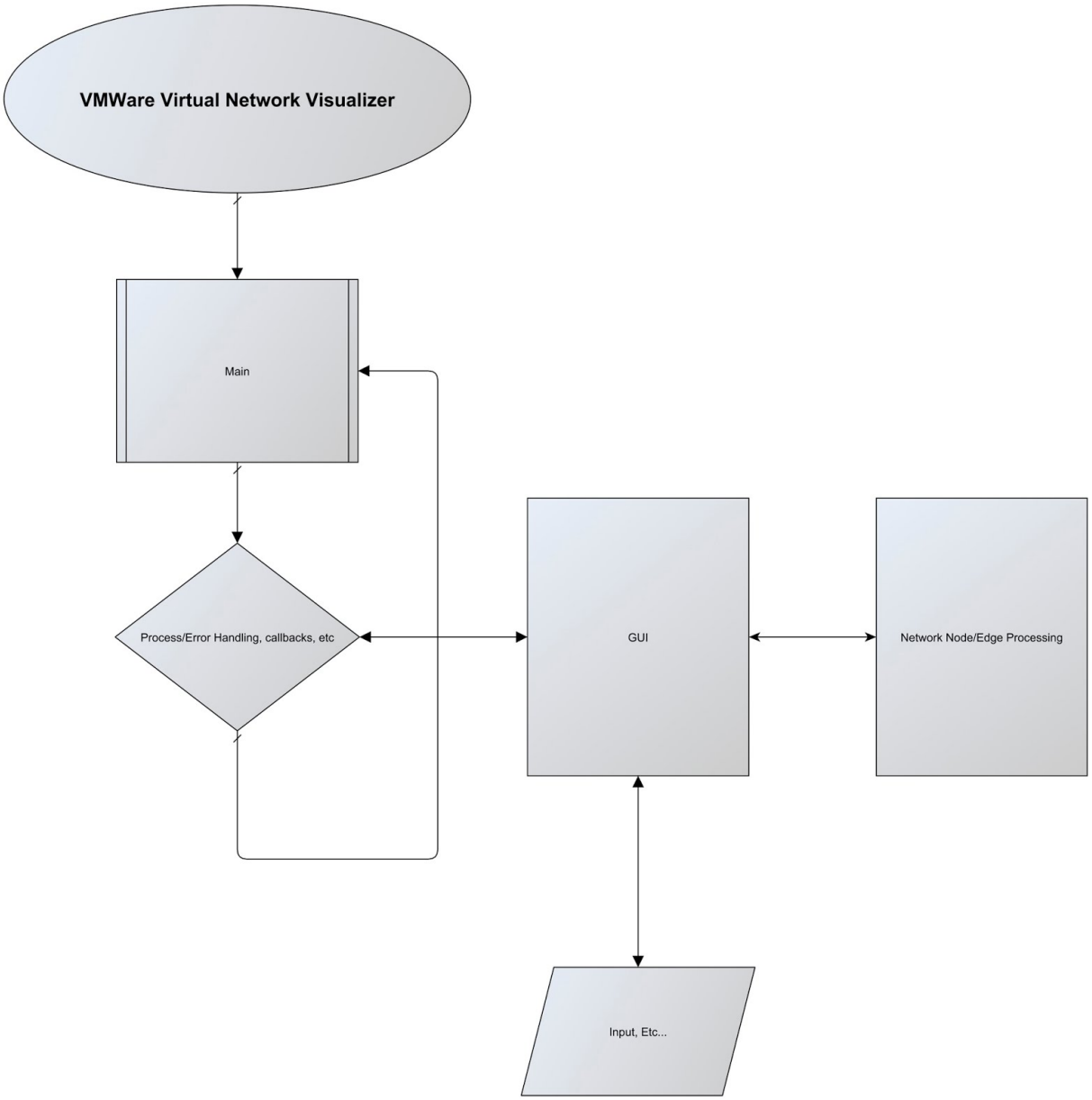
Since this the VMWare Virtual Network Visualizer was created in Java it is very portable. Any computer running with JRE 7 or higher will be able to run this product.

3.5 Design Constraints

A few design limitations arise from the constraints of using the JUNG graphing library. Many features are hidden inside scores of classes, and documentation regarding most of these classes are sparse. What little documentation that can be found on the JUNG library are for the more basic operations, requiring a dive through large amounts of code in order to discover the less trivial features of this library. This process no doubtingly leads to misunderstanding or even not discovering features that can provide a more friendly user experience.

4. Analysis Models

4.1 Data Flow Diagrams (DFD)



5. Change Management Process

At any time any team member may update the SRS document. It is shared by each team member through Google Docs, which retains all revisions made to the document. Once updated it is that team members responsibility to let the other team members know of the changes. Before final revisions are complete, revision histories are overlooked and approved by team collectively before being submitted as a complete revision.

This Page Intentionally Left Blank