Social Networks Visualizer

October 4, 2012

Software Requirements Specification

Version 1.0

Evangelos Motesnitsalis
Undergraduate Student
Department of Informatics
Faculty of Natural Sciences
Aristotle University of Thessaloniki
(A.E.M : 1937)

Prepared for Software Engineering Course

Instructor: Ioannis Stamelos, Associate Professor

Autumn 2012

Revision History

Date	Description	Author	Comments
04/10/2012	Version 1	Evangelos Motesnitsalis	First Revision

Document Approval

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

Signature	Printed Name	Title	Date
	Dimitris Kalamaras	Lead Software Eng.	04/10/2012
	Evangelos Motesnitsalis	Undergraduate Informatics Student	04/10/2012

Table of Contents

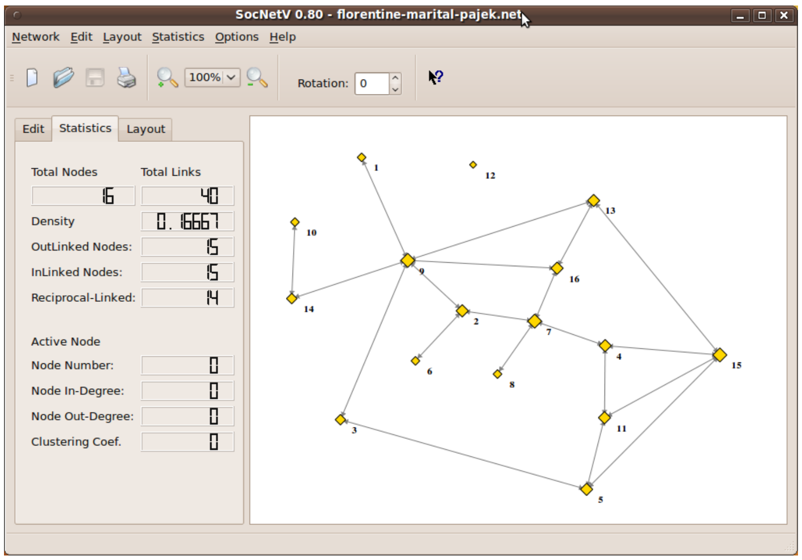
1. Introduction

1.1. Purpose

This Software Requirements Specification provides a complete description of all the functions and specifications of the program “Social Networks Visualizer”.

The expected audience of this document is the department of Informatics of the Faculty of Natural Sciences at the Aristotle University of Thessaloniki as well as the developer and anyone who intends to develop on this program.

1.2. Scope


SocNetV 0.80 - florentine-marital-pajek.net
Network Edit Layout Statistics Options Help
100%
Edit Statistics Layout
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Rotation: 0

Total Nodes	16
Total Links	40
Density	0.16667
OutLinked Nodes:	15
InLinked Nodes:	15
Reciprocal-Linked:	14

Active Node

Node Number:	0
Node In-Degree:	0
Node Out-Degree:	0
Clustering Coef.	0

Social Networks Visualizer (SocNetV) is a flexible and user-friendly tool for the analysis and visualization of Social Networks. It lets you construct networks (mathematical graphs) with a few clicks on a virtual canvas or load networks of various formats (GraphViz, GraphML, Adjacency, Pajek, UCINET, etc) and modify them to suit your needs. SocNetV also offers a built-in web crawler, allowing you to automatically create networks from all links found in a given initial URL.

The application can compute basic network properties, such as density, diameter and distances (shortest path lengths), as well as more advanced structural statistics, such as node and network centralities (i.e. closeness, betweeness, graph), clustering coefficient, etc. Various layout algorithms (i.e. Spring-embedder, radial and layered according to node centralization) are supported for meaningful visualizations of your networks. Furthermore, random networks (Erdos-Renyi, Watts-Strogatz, ring lattice, etc) can be created with a few clicks.

SocNetV is a work in progress and is being developed in C++ and Qt. Our primary target platform is Linux, but you can compile and run SocNetV on OS X and Windows as well, as long as you have Qt4 libraries installed.

The program is Free Software, licensed under the GNU General Public License 3 (GPL3).

1.3. Definitions, Acronyms, and Abbreviations

Term	***Definition***
SocNetV	The software under analysis
IEEE	Institute of Electrical and Electronic Engineers
SRS	Software Requirements Specification
SNA	Social Network Analysis
Padgett's Florentine families	Mathematical Tools

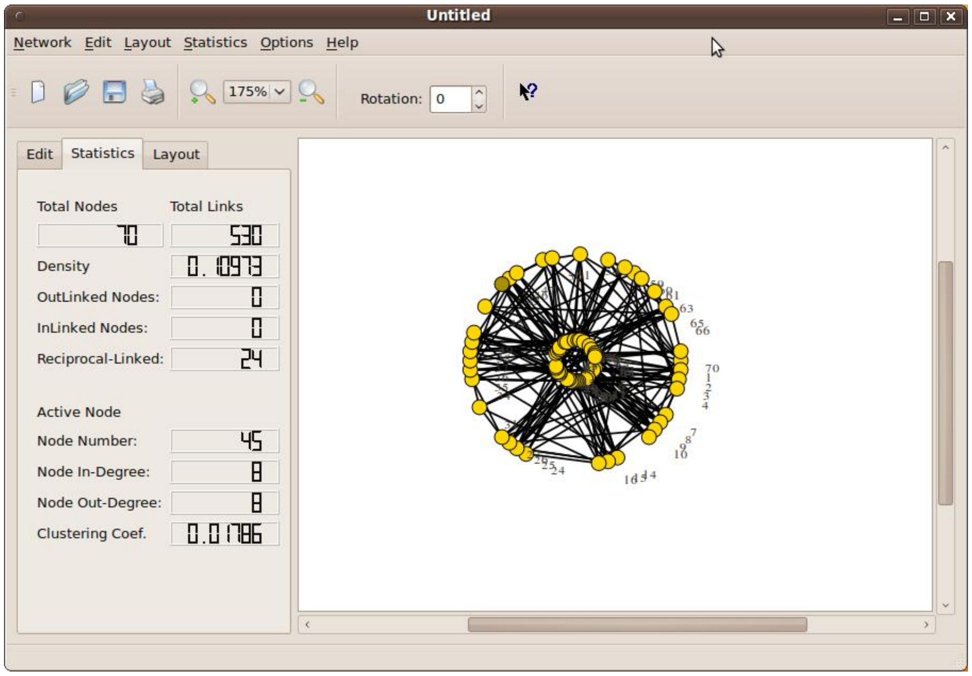
Social Network Analysis

A Social Network is the social structure which facilitates communication between a group of actors (individuals or organizations) that are related somehow (i.e. by common interests, shared values, financial exchanges, friendship, dislike, etc). For instance, your friends and you form a social network. But, social networks operate on many more levels, from family relations and disease spreading up to the level of company strategies, social movements or even nations. Furthermore, research in many scientific areas has shown that social networks are important when we study the way problems are solved, diseases are spreaded, organizations are run, and the degree to which individuals succeed in achieving their goals.

Social Network Analysis (SNA) is a beautiful blend of Sociology and Mathematics, composed of various interdisciplinary techniques for the study of such social networks. SNA researchers conceptualize social relationships in terms of nodes and edges (links) in mathematical graphs. Nodes represent the individual actors within the networks, while edges visualize the relationships between those actors. The result is graph-based structures which are often very complex.

Distance

The geodesic distance is the length of the shortest path between two connected nodes.



Total Nodes: 70
Total Links: 530
Density: 0.10973
OutLinked Nodes: 0
InLinked Nodes: 0
Reciprocal-Linked: 24

Active Node
Node Number: 45
Node In-Degree: 8
Node Out-Degree: 8
Clustering Coef. 0.01786

Distance Matrix

The 'Distance Matrix' option calculates and displays two matrices; the first is the matrix of distances and the second (sigma) the matrix of the number of shortest paths between any two nodes. The latter is used in Centralities calculation.

Graph Diameter

The diameter of a network is the maximum length of all shortest paths between any two connected nodes.

Clustering Coefficient

The Clustering Coefficient of a node quantifies how close the node and its neighbors are to being a clique. This is used to determine whether a network is a small-world or not.

1.4. References

[IEEE] The applicable IEEE standards are published in "IEEE Standards Collection," 2001 edition.
[Bruade] The principal source of textbook material is "Software Engineering: An Object-Oriented Perspective" by Eric J. Bruade (Wiley 2001).
[Book] Teach book "Βασικές Αρχές Τεχνολογίας Λογισμικού", by Sommerville, 8 English edition, copyright 2009 publications Κλειδάριθμος.
[Journal] 830-1984, IEEE Guide to Software Requirements Specifications

1.5. Document overview

The remainder of this document is three chapters, the first providing a full description of the project. It lists all the functions performed by the system. In the third chapter there are some details about the system functions and actions in full for the software developers' assistance.

2. General Description

Social Networks Visualizer (SocNetV) is a project to build a flexible and user-friendly, cross-platform tool for the analysis and visualization of social networks, targeting primarily the researcher. SocNetV lets you construct social networks with a few clicks on a virtual canvas or load networks of various formats (GraphML, GraphViz, Adjacency (Sociomatrix), Pajek, UCINET, etc.) and modify them to suit your needs.

The application can compute all the basic network properties, such as graph diameter, and distances (shortest path lengths), as well as more advanced structural statistics, such as node and network centralities (i.e. Closeness, Betweeness, Graph, etc.), clustering coefficient, etc.

Various layout algorithms (i.e. Energy-based, in circles and in levels according to various centrality indexes) are supported for meaningful visualizations of your networks. Furthermore, random networks (Erdos-Renyi, Watts-Strogatz, ring lattice, etc.) can be created with a few clicks.

SocNetV is a work in progress and is being developed in C++ and Qt, an open-source GUI development toolkit from Nokia. Its primary target platform is Linux, but it can be compiled and run on OS X and Windows as well, as long as the Qt4 libraries are installed.

2.1 Product Perspective

There are no popular related products for Social Network Visualizer.

2.2 Product Functions

Network creation

To start working with SocNetV you need network data, i.e. a graph of nodes (vertices) and links (edges). SocNetV enables the creation of such networks or loads them from files. There are multiple ways to create or edit nodes and links in SocNetV: from the menus, from the dock buttons, or by right/left/middle-clicking on the canvas.

Creating a new node

To create a new node, double-click on the canvas or click on the "Add node" button. The keyboard shortcut is Ctrl+A.

You can move a node by left-clicking on it and moving the mouse.

Creating a new link

To create a new link, middle-click on the source node and then middle-click again on the target node. By default, all links created this way are weighted 1. If your mouse doesn't have middle button , or you find it difficult, you can right-click on the source node, then select "Create Link". In the dialog, just enter the target node number and the desired link weight. Alternatively, you can click on the "Add link" button from the dock. In that case, you will be asked for both the source and the target node numbers (and the link weight). The keyboard shortcut for this action is Ctrl + L.

Link Creation Example: Say you created two nodes, numbered 1 and 2, on the canvas. To create a new link from node 1 to node 2, middle click on node 1 (the mouse pointer will become a hand) and afterwards middle-click on node 2. A new link will be drawn instantly. If you want an edge (double link) repeat the process from node 2 to node 1.

Each link you create this way has the default weight 1 and black color.

Interaction and Group Selection

As mentioned earlier, you move any node by left-clicking and dragging it. If you want to select more than one node, press and hold down the left mouse button on the canvas. By moving your mouse, a rectangle will be drowned. All nodes inside this rectangle will be selected the moment you release the mouse button.

Node Menu and Node Shapes

When you right-click on a node, a context menu appears. There you can remove the node, change its color, label, size as well as its shape. A similar menu appears when you right click on a link.

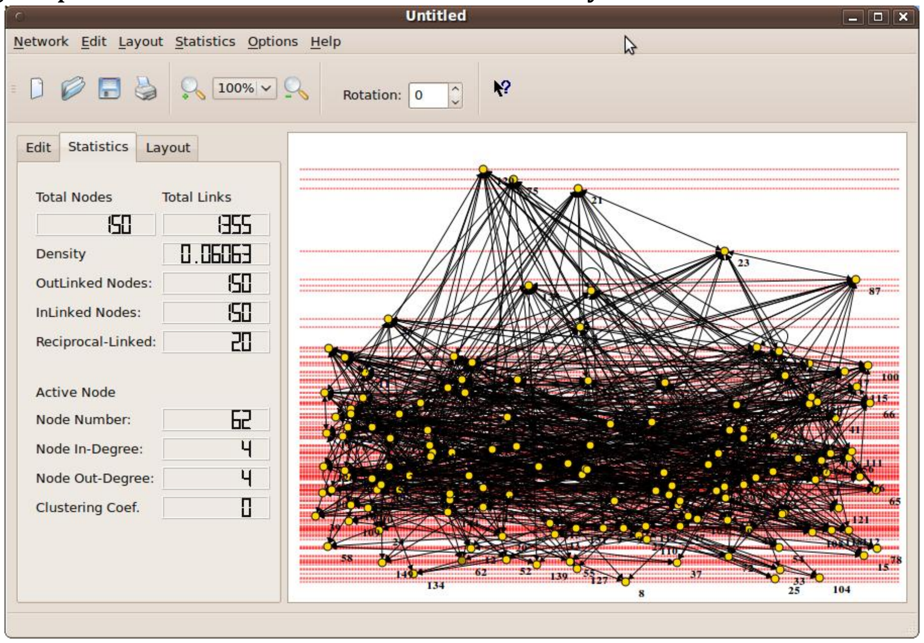
SocNetV supports many kinds of node shapes, i.e. rectangles, diamond, ellipse, circle, etc. To change the shape of a node, right-click on it and in the context menu selects Options and then "Change shape to".

Loading a network

The easiest way to start working with SocNetV is when you have already a network in a supported format (see Formats).For instance, you might have another program (for example a simulation) creating adjacency networks which you want to visualize. In that case, from the SocNetV 's menu go File and then Load. In the dialogue that will appear, navigate to the desired folder and select the appropriate network file. SocNetV will automatically recognize the format and, if it is supported, it will visualize the network.

Saving the active network

To save the active network, just press Ctrl + S or click on the menu entry File and then Save. By default, it will be saved in GraphML format.If you like, you can export it to another supported format (menu Network and then Export To). Note that some formats are supported only for loading - not for saving.



Rotation: 0

Total Nodes	150
Total Links	1355
Density	0.06063
OutLinked Nodes:	150
InLinked Nodes:	150
Reciprocal-Linked:	20

Active Node

Node Number:	62
Node In-Degree:	4
Node Out-Degree:	4
Clustering Coef.	0

View the adjacency matrix

The adjacency matrix of a network is a matrix where each element a (i, j) is equal to the weight of the link from node i to node j. If the nodes are not connected, then a (i, j) = 0.

To view the adjacency matrix of a network, press F6. By default, SocNetV displays the adjacency matrix as integer-valued only (although we do allow float weights).

Random network creation

SocNetV can create a random network for you. It can create the following types of random networks:

Small Worlds: According to the Watts and Strogatz model, a 'small world' is a random network with short average path lengths and high clustering. From the menu Network select Create Random Network > Small World (or press Shift + W). You will be asked for the number of nodes, their initial degree and a rewiring probability.

Erdos - Renyi networks: According to G(n, p) model (Erdos - Renyi), a random network is created by connecting nodes randomly. Each edge is included in the graph with equal probability p, independently of the other edges. From the menu Network select Create Random Network > Erdos-Reny (or press Shift + R). You will be asked for the number of nodes and a link probability.

Ring lattices: Ring lattices (or physicist's lattices) are 'random' networks where all nodes are positioned in a ring and each one has the same even degree (number of links) L with her "neighborhood", namely she is linked with the L/2 nodes before and L/2 nodes after her. For instance in a lattice of 4-lattice of 10 nodes, node 6 will be linked with 4,5,7,8. To create such a network does Network then Create Random Network and then Lattice (or press Shift + L). You will be asked for the number of nodes and the number of links each node will have.

Same degree networks: these are random networks where each node has the same degree but is arbitrarily linked with other nodes (not just the neighbors).

Web Crawler

SocNetV offers a built-in web crawler, allowing you to automatically create networks from all links found in a given website.

A Web Crawler is a software bot (an algorithm), which starts with a given URL (website or webpage) to visit.

As the algorithm crawls that webpage, it identifies all the links in the page and adds them to a list of URLs (called frontier). Then, all the URLs from the frontier are recursively visited.
To start the web crawler, go to menu Network and then Web Crawler or press Shift + C. A dialog will appear, where you must enter initial web address (seed), the maximum recursion level (how many URLs from the frontier will be visited) and the maximum running time.

Printing and Exporting

To print the network directly to your printer, press Ctrl + P.
Keep in mind, that SocNetV follows the "what you see is what you print" principle: print what is viewable in the canvas, i.e. if you zoom-in to a network, the application will only print that specific network portion. So, you might need to zoom-out enough so that the whole network is viewable and therefore printable.
Except printing, you can export your work into raster (BMP and PNG) images, as well as PDF documents. The latter are vector-based, and therefore offer the best quality.

2.3 User Characteristics

Any user familiar with Social Network Analysis and basic concepts of using a personal computer is normally able to use the program. As a result, no specific requirements are affected by the user's characteristics.

2.4 General Constraints

The obvious and most important limitation for using this application is the amount of memory, located inside the computer on which the installation will take place as well as the operating system and the RAM and CPU speed rates.

In addition, the amount of nodes and the complexity of the graph that will be displayed is limited by the screen width and resolution.

.

2.5 Assumptions and Dependencies

There are no dependencies regarding the operating system that the development and the installation will take place.

However, there are dependencies base on the screen resolution as well as the graphics display drivers. There are no other obvious dependencies.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

None important.

3.1.2 Hardware Interfaces

The program must be able to be executed in a simple computer or notebook or netbook.

3.1.3 Software Interfaces

The program is going to need an operating system in order to be installed. It is designed to run in any well known Operating System, including Windows, Linux and Mac OS.

3.1.4 Communications Interfaces

None important.

3.2 Functional Requirements

3.2.1 Node Addition Requirement

The program must allow the user to add a specific node inside a graph.

3.2.1.1 Details

This requirement is one of the very first that will appear during the program execution

3.2.1.2 Inputs

None.

3.2.1.3 Outputs

A node inside the social graph.

3.2.2 Link Addition Requirement

The program must allow the user to add a specific link inside a social graph.

3.2.2.1 Details

This requirement is also one of the very first that will appear during the program execution

3.2.2.2 Inputs

None.

3.2.2.3 Outputs

A link inside the social graph.

3.2.3 Link Removal Requirement

The program must allow the user to remove a specific link from the social graph.

3.2.3.1 Details

This requirement is also one of the very first that will appear during the program execution

3.2.3.2 Inputs

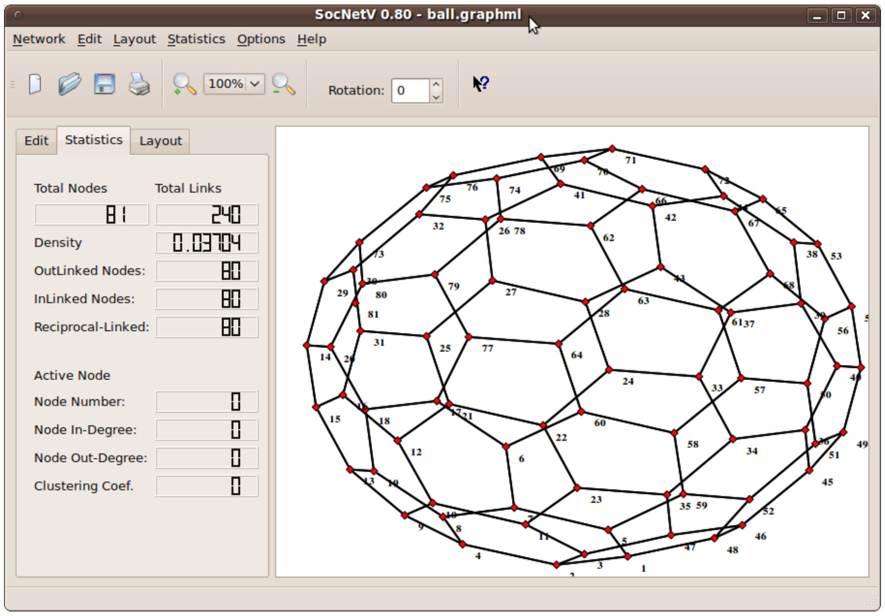
The desired link.

3.2.3.3 Outputs

The previous graph without the link that was selected to be removed.

3.2.5 Display Statistics Requirement

The program must allow the user to display the social network's statistics data.



Total Nodes	81
Total Links	240
Density	0.03704
OutLinked Nodes:	80
InLinked Nodes:	80
Reciprocal-Linked:	80
Node Number:	0
Node In-Degree:	0
Node Out-Degree:	0
Clustering Coef.	0

3.2.5.1 Details

This requirement is one of the most important during the results phase of the execution of this program.

3.2.5.2 Inputs

The desired graph.

3.2.5.3 Outputs

All the possible statistics like:

1. Total Nodes
2. Total Links
3. Density
4. OutLinked Nodes
5. InLinked Nodes
6. Reciprocal-Linked

7. Network Symmetry
8. Graph Distance
9. Diameter
10. Number of Cliques
11. Clustering Coefficient
12. Triad Census
13. Node Centralities etc.

3.2.6 Print Statistics Requirement

The program must allow the user to print the social network's statistics data.

3.2.6.1 Details

This requirement is one of the most important during the results phase of the execution of this program.

3.2.6.2 Inputs

The desired graph.

3.2.6.3 Outputs

All the possible statistics, printed in paper through a printer, like:

1. Total Nodes
2. Total Links
3. Density
4. OutLinked Nodes
5. InLinked Nodes
6. Reciprocal-Linked
7. Network Symmetry
8. Graph Distance
9. Diameter
10. Number of Cliques
11. Clustering Coefficient
12. Triad Census
13. Node Centralities etc.

3.2.7 Print Network Requirement

The program must allow the user to print the social network itself.

3.2.7.1 Details

This requirement is one of the most important during the results phase of the execution of this program.

3.2.7.2 Inputs

The desired graph.

3.2.7.3 Outputs

The social network, printed in paper through a printer, exactly like the way it is displayed in the screen.

3.2.8 Export Network Requirement

The program must allow the user to export the social network itself.

3.2.8.1 Details

This requirement is one of the most important during the results phase of the execution of this program. It is crucial because the output file may be used from another program in the future.

3.2.8.2 Inputs

The desired graph.

3.2.8.3 Outputs

A file that includes the social network, in the selected format.

3.2.9 Open File Requirement

The program must allow the user to open a file that contains social network data.

3.2.9.1 Details

This requirement is one of the most important aspects of the execution of this program. It is crucial because its absence may lead to an abnormal functioning program.

3.2.9.2 Inputs

The desired file path.

3.2.9.3 Outputs

The desired file's data, displayed in the program window.

3.2.10 Save File Requirement

The program must allow the user to save a file that contains social network data.

3.2.10.1 Details

This requirement is one of the most important aspects of the execution of this program. It is crucial because its absence may lead to an abnormal functioning program.

3.2.10.2 Inputs

The desired file.

3.2.10.3 Outputs

The desired file, altered by its new version.

3.2.11 Close File Requirement

The program must allow the user to close a file that contains social network data.

3.2.11.1 Details

This requirement is one of the most important aspects of the execution of this program. It is crucial because its absence may lead to an abnormal functioning program.

3.2.11.2 Inputs

The desired file.

3.2.11.3 Outputs

The program window without the previously opened file.

3.2.12 Create Random Network Requirement

The program must allow the user to create a network from random data.

3.2.12.1 Details

This requirement is important when there are no available network data

3.2.12.2 Inputs

User's choices like number of edges or number of nodes.

3.2.12.3 Outputs

A randomly created network limited by the input user choices.

3.2.13 Create Random Network Requirement

The program must allow the user to create a network from random data.

3.2.13.1 Details

This requirement is important when there are no available network data

3.2.13.2 Inputs

User's choices like number of edges or number of nodes.

3.2.13.3 Outputs

A randomly created network limited by the input user choices.

3.2.14 View Adjacency Matrix Requirement

The program must allow the user to view every graph's adjacency matrix.

3.2.14.1 Details

This requirement is important for giving the user mathematical data based on the displayed social network.

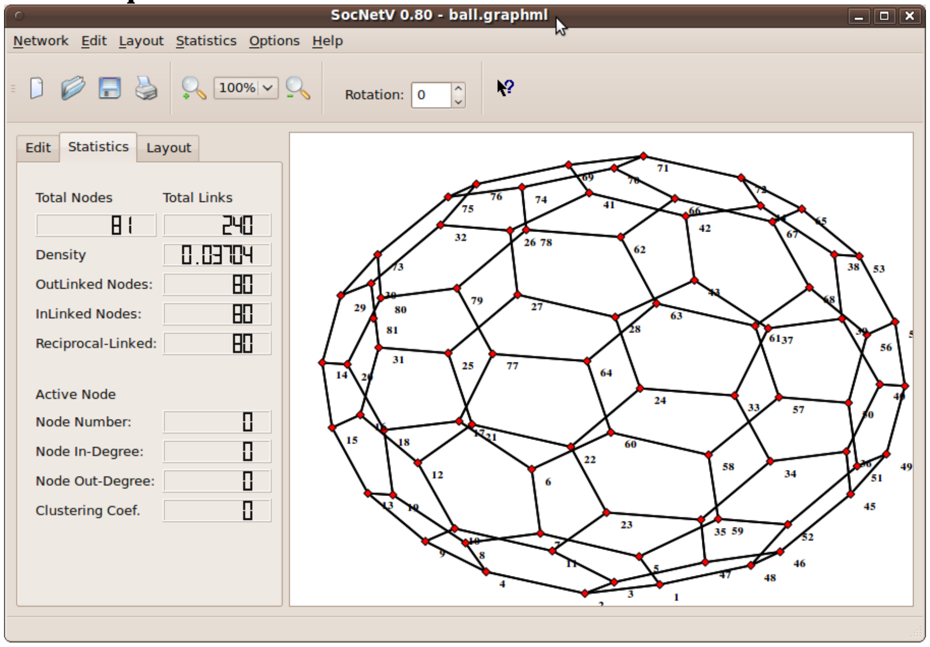
3.2.14.2 Inputs

The desired graph.

3.2.14.3 Outputs

An adjacency matrix that was created from the input's desired graph.

3.2.15 Link Symmetrization Requirement



Total Nodes 81
Total Links 240
Density 0.03704
OutLinked Nodes: 80
InLinked Nodes: 80
Reciprocal-Linked: 80

Active Node
Node Number: 0
Node In-Degree: 0
Node Out-Degree: 0
Clustering Coef. 0

Rotation: 0

The program must allow the user to symmetrize the links of every graph's visual content.

3.2.15.1 Details

This requirement is important for giving the user a better view and understanding of the network's visual appearance.

3.2.15.2 Inputs

The desired graph.

3.2.15.3 Outputs

The previous graph with symmetrized links.

3.2.16 Graph Rotation Requirement

The program must allow the user to rotate the under process graph.

3.2.16.1 Details

This requirement is important for giving the user a better view and understanding of the network's visual appearance.

3.2.16.2 Inputs

The desired graph and the desired angle.

3.2.16.3 Outputs

The previous graph rotated by the given angle.

3.2.17 Colors Change Requirement

The program must allow the user to change any color of any part of the under process graph.

3.2.17.1 Details

This requirement is important for giving the user a better view and understanding of the network's visual appearance as well as providing a better and more user-friendly work environment.

3.2.17.2 Inputs

The desired graph and the desired colors.

3.2.17.3 Outputs

The previous graph, changed by the colors given in the input.

3.2.18 Open Web Crawler Requirement

The program must allow the user to open and use the web crawler.

3.2.18.1 Details

This requirement is important for giving the user the opportunity to use the web in order to take webpage data that can be used in the program.

3.2.18.2 Inputs

A URL.

3.2.18.3 Outputs

The data taken from the algorithms that were applied to the given URL.

3.2.19 Provide Help Requirement

The program must provide a help section to the user.

3.2.19.1 Details

This requirement is important for giving the user some FAQs as well as some “tips of the day” and “about” sections.

3.2.19.2 Inputs

None.

3.2.19.3 Outputs

Answered questions, develepoper’s data, Tips e.t.c.

3.2.20 Create Known Datasets Requirement

The program must be able to automatically create known data sets, such as Padgett’s Florentine families etc.

3.2.20.1 Details

This requirement is important for giving the user mathematical arrays that have quite interesting results in social networks analysis.

3.2.20.2 Inputs

The desired selection among a specific list.

3.2.20.3 Outputs

The created dataset, based on the input selection.

3.2.21 Display Selected Node’s Statistics Requirement

The program must be able to display a specific node’s statistics to the user.

3.2.21.1 Details

This requirement is important for giving the user an overview of each node that the network is consisted of.

3.2.21.2 Inputs

The desired node.

3.2.21.3 Outputs

Node's statistics like:

- Node number
- Node In-degree
- Node Out-degree
- Clustering Coefficient

3.2.22 Change Node Size Requirement

The program must be able to change a node's size according to the user's needs.

3.2.22.1 Details

This requirement is important for giving the user the ability to highlight the important nodes inside a network.

3.2.22.2 Inputs

The desired mode (indegree/outdegree).

3.2.22.3 Outputs

The previous network, changed by the new nodes sizes that were selected in the output.

3.2.23 Automatically Change Layout Requirement

The program must be able to change the layout of the under analysis network in an automatic way.

3.2.23.1 Details

This requirement is important for giving the user the ability to take the network to an accepted form.

3.2.23.2 Inputs

None.

3.2.23.3 Outputs

The previous network, changed by the algorithm that the program uses to change layout.

3.2.24 Manually Change Layout Requirement

The program must be able to give the user the ability to change the layout of the under analysis network.

3.2.24.1 Details

This requirement is important for giving the user the ability to take the network to an accepted form.

3.2.24.2 Inputs

The desired network and the specific change to be done.

3.2.24.3 Outputs

The previous network, changed in the way the users desires.

3.2.25 Graph Rotation Requirement

The program must allow the user to zoom in or out the under process graph.

3.2.25.1 Details

This requirement is important for giving the user a better view and understanding of the network's visual appearance.

3.2.25.2 Inputs

The desired graph and the desired zoom.

3.2.25.3 Outputs

The previous graph zoomed by the given number.

3.2.26 Text Editor Opening Requirement

The program must allow the user to open a text editor.

3.2.26.1 Details

This requirement is important for giving the user the ability to manually change a data file and reload it to the network visualizer.

3.2.26.2 Inputs

None.

3.2.26.3 Outputs

The text editor window.

3.2.27 Node Edit Requirement

The program must allow the user to edit a node’s properties.

3.2.27.1 Details

This requirement is important for giving the user the ability to manually change a specific node and maintain a better control of the whole network. The properties are: the color the size, the value and the label

3.2.27.2 Inputs

The desired node and the desired properties data.

3.2.27.3 Outputs

The desired node, changed by the properties given in the input.

3.2.28 Nodes Edit Requirement

The program must allow the user to edit the whole amount of nodes inside a network.

3.2.28.1 Details

This requirement is important for giving the user the ability to manually change the whole network and maintain a better control of it. The properties are: the color the size, the value and the label

3.2.28.2 Inputs

The desired properties data.

3.2.28.3 Outputs

The previous graph, changed by the properties given in the input.

3.2.29 Edge Edit Requirement

The program must allow the user to edit an edge’s properties.

3.2.29.1 Details

This requirement is important for giving the user the ability to manually change a specific edge and maintain a better control of the whole network. The properties are: the color the size, the value and the label

3.2.29.2 Inputs

The desired edge and the desired properties data.

3.2.29.3 Outputs

The desired edge, changed by the properties given in the input.

3.2.30 Nodes Edit Requirement

The program must allow the user to edit the whole amount of edges inside a network.

3.2.30.1 Details

This requirement is important for giving the user the ability to manually change the whole network and maintain a better control of it. The properties are: the color the size, the value and the label

3.2.30.2 Inputs

The desired properties data.

3.2.30.3 Outputs

The previous graph, changed by the properties given in the input.

3.2.31 Layout Edit Requirement

The program must allow the user to edit the layout of a network in multiple ways.

3.2.31.1 Details

This requirement is important for giving the user the ability to manually change the whole network and maintain a better control of it. Changes in the layout focus on Centrality, Betweeness, InDegree, OutDegree and others.

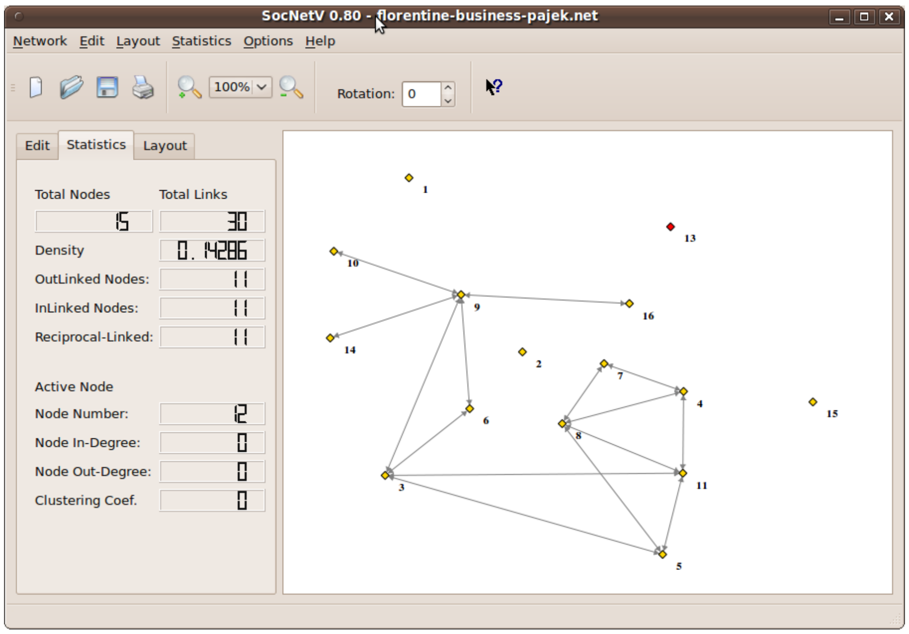
3.2.31.2 Inputs

The desired layout change data.

3.2.31.3 Outputs

The previous graph, changed by the given layout data.

3.2.32 Display Edit Requirement



SocNetV 0.80 - florentine-business-pajek.net

100%

Rotation: 0

Total Nodes	Total Links
15	30

Density: 0.14286
OutLinked Nodes: 11
InLinked Nodes: 11
Reciprocal-Linked: 11

Active Node
Node Number: 12
Node In-Degree: 0
Node Out-Degree: 0
Clustering Coef. 0

The program must allow the user to edit the displayed properties of a network in multiple ways.

3.2.32.1 Details

This requirement is important for giving the user the ability to manually change the whole network and maintain a better control of it. The possible display properties are: node number, edge number, edge arrows and others.

3.2.32.2 Inputs

The desired display change data.

3.2.32.3 Outputs

The previous graph, changed by the given display properties.

3.2.33 View Edit Requirement

The program must allow the user to edit the view of a network in multiple ways.

3.2.33.1 Details

This requirement is important for giving the user the ability to come to a better understanding of the network as well as provide a more user-friendly environment. This means that it must provide ways of changing the program's window background or add/remove toolbars.

3.2.33.2 Inputs

The desired view change data.

3.2.33.3 Outputs

The previous window, changed by the given view properties.

3.3 Non-Functional Requirements

3.3.1 Performance

The display time in changes must be done quickly and efficiently. They must not exceed the 1-2 seconds for big networks.

3.3.2 Reliability

Reliability in this program must be focused in advanced topics. The algorithms that uses must provide correct results even for big amounts of data.

3.3.3 Availability

There are no high concerns about availability in this program. If for any reason the program makes itself not available for some moments, this event is not going to affect the user in any important way.

3.3.4 Security

There are no security requirements for this software.

3.3.5 Maintainability

The program must be designed in a way that new addition can be done without changing the already developed structure.

3.3.6 Portability

The program is completely portable. There is no need for installation for this software.

3.4 Inverse Requirements

There are no useful inverse requirements for this product.

3.5 Design Constraints

No worth-mentioning design constraints apart from the GNU License that this product development was based.

4. Change Management Process

Requirements evolve throughout the project, especially as described by the specific practices of the Requirements Development process area and the Technical Solution process area. As the requirements evolve, this specific practice ensures that project participants commit to the current, approved requirements and the resulting changes in project plans, activities, and work products. Regarding the post-release phase of this project the change management process for this specific software consists of the following phases.

- The project participants (users/developers/testers) find the requirement under process/change
- Create a report that is submitted to the developer and possibly the whole development team.
- Specify the exact impact caused by the change.
- The requirement goes under discussion/evaluation around the whole team.
- The developers decide if the change is applicable.
- The writer of the SRS documentation proceeds with change and adds a record in the change specification list.
- The developers proceed with the actual code implementation.

4.1 Requirements Decision Database

Subpractice 1: Document all requirements and requirements changes that are given to or generated by the project.

Subpractice 2: Maintain the requirements change history with the rationale for the changes. Maintaining the change history helps track requirements volatility.

Subpractice 3: Evaluate the impact of requirement changes from the standpoint of relevant stakeholders.

Subpractice 4: Make the requirements and change data available to the project.

4.2 Requirements tracking system

Subpractice 1: Maintain requirements traceability to ensure that the source of lower level (derived) requirements is documented.

Subpractice 2: Maintain requirements traceability from a requirement to its derived requirements and allocation to functions, interfaces, objects, people, processes, and work products.

Subpractice 3: Generate the requirements traceability matrix.

A. Appendices

A.1 Appendix 1 – Supported Formats

SocNetV supports many network formats:

1. GraphML (.graphml),
2. GraphViz (.dot),
3. Adjacency matrix (.net, .txt)
4. Pajek-like (.net),
5. UCINET's Data Language (.dl)

You can load these kinds of files by clicking on menu File > Load or by specifying them explicitly at the command line. SocNetV uses simple inspection routines to check the format of the given file. In most cases, it will load the file, no matter what the file extension is.

WARNING: The default file format of SocNetV is GraphML. If you create a new network and press Ctrl + S to save it, then by default SocNetV will save it in GraphML format.

1. **GraphML files**

Each GraphML document is written in a special form of XML and defines a graph. For instance the code below, contains 11 nodes and 12 edges:

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
<graph id="G" edgedefault="undirected">
<node id="n0"/>
<node id="n1"/>
<node id="n2"/>
<node id="n3"/>
<node id="n4"/>
<node id="n5"/>
<node id="n6"/>
<node id="n7"/>
<node id="n8"/>
```

```
<node id="n9"/>
<node id="n10"/>
<edge source="n0" target="n2"/>
<edge source="n1" target="n2"/>
<edge source="n2" target="n3"/>
<edge source="n3" target="n5"/>
<edge source="n3" target="n4"/>
<edge source="n4" target="n6"/>
<edge source="n6" target="n5"/>
<edge source="n5" target="n7"/>
<edge source="n6" target="n8"/>
<edge source="n8" target="n7"/>
<edge source="n8" target="n9"/>
<edge source="n8" target="n10"/>
</graph>
</graphml>
```

All GraphML files consist of a graphml element and a variety of subelements: graph, node, edge, keys. SocNetV understands all of them.

Nodes are defined by the <node id="n1" /> where id is a unique node identification string. This id is used in edge declaration, below.

Edges are defined by the <edge source="n1" target="n1" /> where source and target are equal to existing node ids.

2. GraphViz files

This is the file format of the graphviz layout package. Unfortunately, I have not yet managed to implement the whole specifications of this nice format. The features that are recognized by SocNetV are displayed in the following example:

```
digraph mydot {
node [color=red, shape=box];
a -> b -> c ->d
node [color=pink, shape=circle];
d->e->a->f->j->k->l->o
[weight=1, color=black];
}
```

Nodes are defined by the "node" declaration. In this you can define the color and the shape of the nodes that will follow. Each link is denoted by an "->" for directed graphs (digraphs) and a "-" for undirected graphs (graphs) between nodes' labels. For instance, "a -> b" means a directed edge from a to b. Moreover, links can have weights and colours.

3. Adjacency Matrix files

The adjacency sociomatrix format is a very easy one. It describes one-mode networks and contains a simple matrix NxN, where N is the amount of nodes. Each (i,j) element is a number.

If (i,j)=0 then nodes i and j are not connected.

If (i,j)=x where x a non-zero number then there will be an arc from node i to node j.

Again, negative weights are allowed. Those are depicted as dashed lines when the network is visualised on the canvas.

This is an example of an adjacency sociomatrix formatted network.

```
0 0 0 0 0 0 0 0 1 1
0 0 0 0 1 0 1 1 0 0
0 0 0 1 1 0 0 0 0 0
0 0 1 0 0 0 0 0 1 0
0 1 1 0 0 0 1 0 0 0
0 0 0 0 0 0 1 1 0 0
0 1 0 0 1 1 0 0 0 1
0 1 0 0 0 1 0 0 0 0
1 0 0 1 0 00 0 0 0
1 0 0 0 0 0 1 0 0 0
```

Two-mode Sociomatrix files

Unlike one-mode networks which describe direct links between actors of the same type, networks can be two-mode as well. Two-mode networks describe either two sets of actors or a set of actors and a set of associated events.

In the first case, which usually is called dyadic two-mode network, there are two sets of actors. The sociomatrix codifies the relations between actors in the first set and actors in the second set.

In the second case, which usually is called affiliation network, there is a set of actors and a set of events or organizations. The sociomatrix measures the attendance or affiliations of the actors (first mode) with a particular event or organization (second mode).

Two-mode networks are described by affiliation network matrices, where A(i,j) codes the events/organizations each actor is affiliated.
A two-mode sociomatrix is a matrix NxM, where N is the amount of nodes and M is the amount of events. Each (i,j) element can be 0 or 1.

If A(i,j)=1 then actor i is affiliated with event j.
This is an example of an two-mode sociomatrix formatted network.

```
0 0 1 1 0 0 0 0 1
0 0 1 0 1 0 1 0 0
0 0 1 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0
0 0 1 1 0 0 0 0 0
0 0 0 1 0 0 1 0 0
1 0 0 1 0 0 0 1 0
0 0 1 0 0 0 0 0 1
0 1 1 0 0 0 0 0 1
0 0 0 1 0 0 1 0 0
0 0 1 1 1 0 0 0 1
```

4. Pajek-like formatted files

Note the 'Pajek-like' part. This is because real Pajek files can be much more complicate than the ones recognized by SocNetV. To be more precise, here is an example of the Pajek-like form that SocNetV understands. The numbers to the left are just indicating line numbers.

```
1) *Network
2) *Vertices 6
3) 1 "pe0" ic LightGreen 0.5 0.5 box
4) 2 "pe1" ic LightYellow 0.8473 0.4981 ellipse
5) 3 "pe2" ic LightYellow 0.6112 0.8387 triangle
6) 4 "pe3" ic LightYellow 0.201 0.7205 diamond
7) 5 "pe4" ic LightYellow 0.2216 0.2977 ellipse
8) 6 "pe5" ic LightYellow 0.612 0.1552 circle
9) *Arcs
10) 1 2 1 c black
11) 1 3 -1 c red
12) 2 4 1 c black
13) 3 5 1 c black
14) *Edges
15) 6 4 1 c black
16) 5 6 1 c yellow
```

The first line (*Network) declares that this is a Pajek network.

The second line (*Vertices 6) declares the number of vertices of the network and identifies that the following lines describe node properties.

Each one of the following 6 lines (3-8) constructs one node. Each node's line has 7 columns-properties:
Column 1 denotes the node's number.
Column 2 denotes the node's label.
Column 3 indicates that the next column carries the color of the node's shape.

Column 4 denotes the color of the node's shape.
Column 5 denotes the proportional X coordinate of the specific node on the canvas.
Column 6 denotes the proportional Y coordinate of the specific node on the canvas.
Column 7 denotes the node's shape.
Line 9 (*Arcs) identifies that the following lines will describe arcs from an node to another. Each one of the lines 10-13 constructs one arc. For instance, Line 10 constructs an arc from node 1 to node 2 with weight 1 and black color.

Line 14 identifies that the following lines will describe edges (double arcs) between nodes. Each one of the lines constructs one edge. For instance, Line 10 constructs an arc from node 1 to node 2 with weight 1 and black color.

Note that it is legal to have mixed columns in Pajek-like network file. For instance you can have a node's specification line like this:

4 "label" 0.201 0.7205 is LightYellow diamond.

Also, it is not necessary to declare X and Y coordinate or colors and shapes. In that case SocNetV will use the defaults, which are red diamonds scattered randomly across the canvas. Nevertheless, the first two columns must be valid node numbers and labels.

Note also that weights might be negative as in line 11. Negative weights are depicted as dashed lines on the canvas.

Color names are not arbitrarily created. Valid color names for nodes and arcs/edges are those specified in the X11 file: /usr/X11R6/lib/X11/rgb.txt, i.e. red, gray, violet, navy, green, etc. You can change colors of all network elements from inside SocNetV.

SocNetV also supports Pajek files which declare edges/arcs in matrices, like this:

```
*Vertices     11
    1 "minister1"                        0.2912    0.2004 ellipse
    2 "pminister"                        0.4875    0.0153 diamond
    3 "minister2"                        0.3537    0.3416 ellipse
    3 "minister2"                        0.3537    0.3416 ellipse
    4 "minister3"                        0.4225    0.5477 ellipse
    5 "minister4"                        0.4538    0.1603 ellipse
    6 "minister5"                        0.4900    0.3836 ellipse
    7 "minister6"                        0.6212    0.5038 ellipse
    8 "minister7"                        0.6450    0.2023 ellipse
    9 "advisor1"                        0.6488    0.6031 box
   10 "advisor2"                         0.3212    0.5515 box
   11 "advisor3"                         0.7188    0.4218 box
*Matrix
 0 1 1 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 1 0 0 0
 1 1 0 1 0 1 1 1 0 0 0
 0 0 0 0 0 0 1 1 0 0 0
 0 1 0 1 0 1 1 1 0 0 0
 0 1 0 1 1 0 1 1 0 0 0
 0 0 0 1 0 0 0 1 1 0 1
 0 1 0 1 0 0 1 0 0 0 1
 0 0 0 1 0 0 1 1 0 0 1
 1 0 1 1 1 0 0 0 0 0 0
 0 0 0 0 0 1 0 1 1 0 0
```

Here, the *Matrix tag replaces *Arcs or *Edges. An ordinary adjacency matrix follows describing all links.

Another possibility, is the *ArcsList tag. When SocNetV finds that tag in a Pajek file, it expects each node to declare list of its link to other nodes. Here is an example:

```
*Vertices 9
1
2
3
4
5
6
```

```
7
8
9
*Arcslist
2 1 3 9
1 3 4 5
3 1 4 7
4 1 2 3
5 1 3 4
7 2 8 9
```

For instance, the first line after *Arcslist means: "node 2 is connected to nodes 1, 3 and 9". It is very simple.

5. UCINET's DL files

UCINET's DL format is one of the easiest to understand. For the moment, we support only FULL MATRIX mode. Each file starts with the "DL" mark; then the amount N of nodes is declared and the format (i.e. if a diagonal is present or not). Then, after the "LABELS:" mark we read the labels of each node line by line. That is, if N was 100 then we expect to read 100 labels. In the end, a DL file declares network data ("DATA") which is only the edges. For instance the network below, contains 4 nodes and 7 arcs/edges:

FORMAT = FULLMATRIX DIAGONAL PRESENT

LABELS:

On the normalization and visualization of author co-citation data: Salton's cosine versus the Jaccard index. Caveats for the use of citation indicators in research and journal evaluations

Should co-occurrence data be normalized? A rejoinder

Home on the range - What and where is the middle in science and technology studies?

```
DATA:
0 0 0.158114 0
0.201234 0 1 0
1 0 0 0
0.1 1 1 0
```