Graphalyzer

May1618

Table of Contents

Team Roles

- Andrew Bowler - Webmaster
- Alberto Gomez-Estrada - Communications
- Michael Sgroi - Key Concept Handler
- Richard White - Key Concept Handler
- Taylor Welter - Team Lead

Problem Statement

Graphs are an elegant way to illustrate the extensive web of connections between many points of data. As the data set becomes larger and more complex, it becomes more important to be able to visualize specific instances of various relationships. The goal of this project is to build a generic graph visualization, analysis, and search web application.

Feasibility

Our team has extensive experience in various technologies, including those of web development and have all taken classes on graph theory. Similar products exist, but this product shall meet the specific needs of Workiva and their clientele.

Risks

Risk	Probability of Occurrence	Criticality (0-100)	Risk Factor (Occurrence * Criticality)	Risk Mitigation Strategy
The project requires more time to develop than expected	0.2	70	14	The group shall meet up frequently to establish our team goals in relation to our deadlines
Project is not able to scale to extremely large data sets	0.5	30	15	We shall try to design the project to load only pieces of the graph at a time. Also the frontend shall only display a certain amount of nodes/edges
Project is inferior to existing products on the market	0.1	20	2	We shall specially tailor the project to meet the needs and requirements of Workiva and its clients
The application doesn't work properly or bugs prevent the application from working	0.3	90	27	We shall test our application with unit tests. And try to design the project to mitigate bugs
Backend language has insufficient support to do what we need	0.1	80	8	We shall verify our needs and check if the language supports them

Table 1: Risks

Proposed Design

The project shall be set up as a client-server model. The client (aka frontend) shall be a web application written in HTML and JavaScript with various libraries, with Angular as the application's front-end framework. It shall be responsible for displaying the graphs and receiving user input. The server (aka the backend) shall be written in D. It shall first be responsible for accepting newly generated graphs from various services via rest. It shall also be responsible for serving up the graph to the frontend. It shall additionally be responsible for performing all the graph calculations and sending the results to the client. Communication between the server and client shall use websockets sending json messages.

Assessment of Proposed Design

Pros

- Using D as a server side language shall provide speed and memory savings.
- Not storing the entire graph shall save us space, since we have no guarantee as to the size of the graph file that shall be loaded onto the application.
- Keeping most of the computation on the server side shall help ensure that the client has a more responsive and efficient experience with our product.
- Using the client/server model with websockets shall allow different frontend to be easily developed in the future.

Cons

- Since D is not as widely used as other server languages, there is no guarantee that there shall be support for necessary features that may arise in the future.
- Deciding which aspects of the graph to maintain will be crucial, since not storing the entire graph creates the possibility of the client needing data that is currently not loaded onto the application.
- If the network speed is slow it may have been faster to do the computations on the client side.

Requirements

User Requirements
The web application must be able to facilitate the analysis of large volumes of data, as well as extracting fragments of data that are most relevant to a client, according to their needs. The web application must be easily accessible to the client, and be fast and efficient. Lastly, because of the volumes of data and the potential for critical data to be analyzed using the application, the application must be thoroughly tested and free of fatal error or complications.

System Requirements
The web application must be deployed on a Linux virtual machine that can be remotely accessed by clients and other users. The application must also process and index large volumes of data efficiently and accurately. The technologies used shall preferably be compatible with technology that is currently in use at Workiva.

Backend Functional Requirements

- The product shall allow for the upload of arbitrary graph files for analysis via REST
- The product shall index incoming information associated with nodes and edges
- The product shall allow for the searching of nodes by name
- The product shall process the graph to determine:
    - Depth
    - Breadth
    - Interconnectedness
    - Impact

Frontend Functional Requirements

- The product shall be able to visualize the graph
    - The graph's visualization shall clearly distinguish different nodes and their connections
    - The visualization shall clearly distinguish properties of the graph through color coding, such as impact and interconnectedness
- The product shall allow for the exploration of the graph via click or touch
- The product shall display the ability to search the graph in the user interface
- The product shall maintain an immutable search history easily accessible to the user for reuse of search terms
- The product shall display to the user when it is processing a user's search query

Nonfunctional Requirements

- The product shall be able to process, visualize parts of, and analyze graph files with sizes up to 32GB
- The product shall not crash or lose connection to the REST service while in use
- The product shall not have any memory leaks or any loss of graph data.
- The product shall not voluntarily or involuntarily modify the graph data.

Minimal Viable Product

At minimum, our web application must be able to display an abstraction of the graph that maintains key information, such as interconnectedness, depth, and impact. The user shall be able to perform cursory searches and store those searches. Finally, search operations performed on the graph shall be stored in an immutable database for security.

The product's most initial development version is estimated for demo around Week 12 of the semester, and the MVP is estimated to be demoed around Week 15 of the semester. During this time period, work will be done in 2-week sprints.

Validation and Acceptance Test

<See Test Plan for more details.>

Future Iterations

Please see the project schedule for more details on work expectations after MVP completion.

Concept Sketch

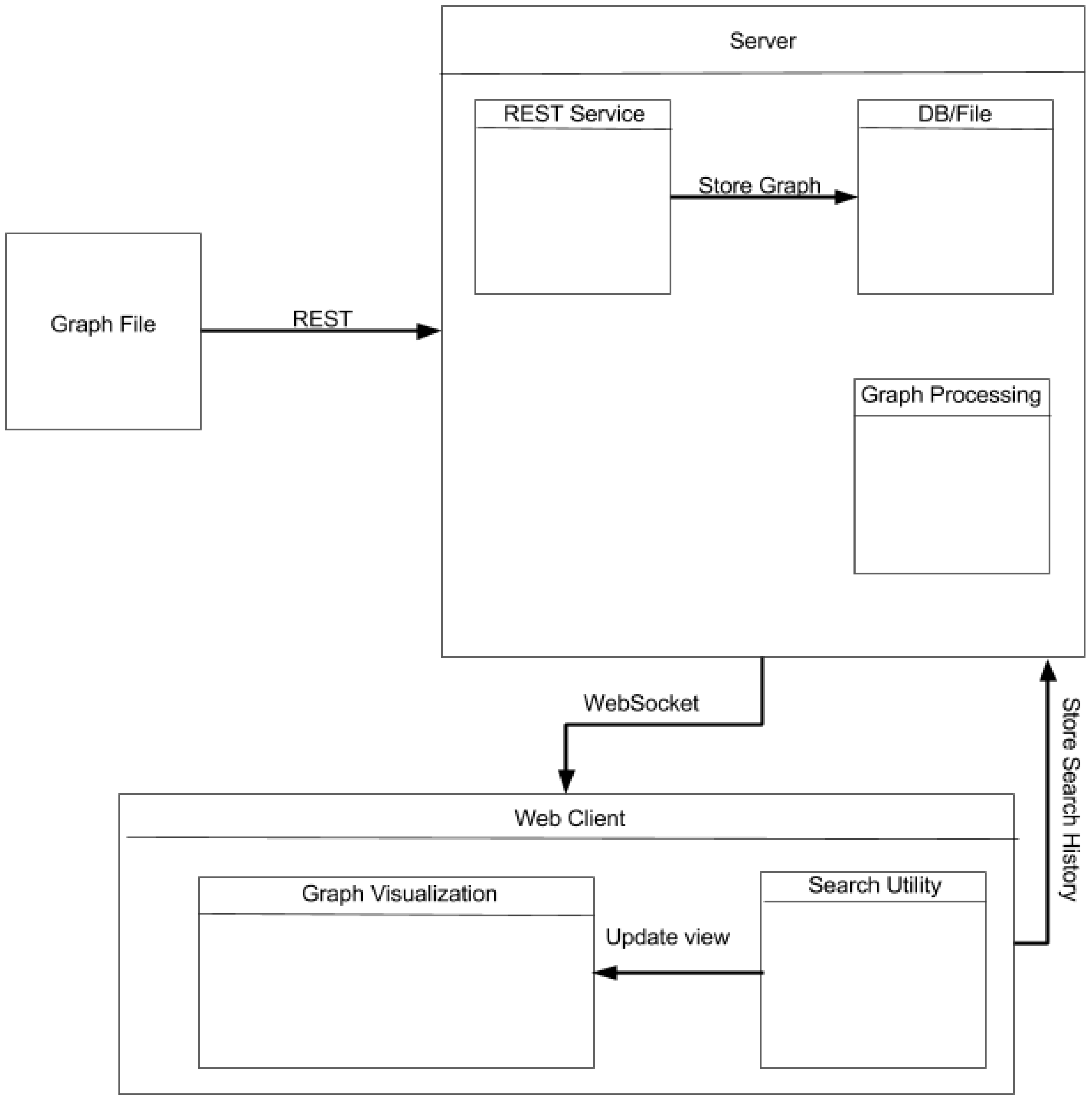

Figure A. Concept Sketch

Interface Description

Further negotiation with Workiva is required for what they expect for a user interface. At the moment, a basic user interface for Graphalyzer shall have a large graph display on the center of the screen, with a search bar at the top. On the side, information for the selected node will be displayed.

Test Plan

Requirement	Validation Test
The product shall allow for the upload of arbitrary graph files for analysis via REST	Upload sample graph files
The product shall index incoming information associated with nodes and edges	Store the information in the database, assert it against the expected outcome
The product shall allow for the searching of nodes by name	Query the name row in the database and confirm it against the search results
The product shall process the graph to determine: ■ Depth ■ Breadth ■ Interconnectedness ■ Impact	Unit tests based on these various qualities of graphs
The product shall be able to visualize the graph ■ The graph's visualization shall clearly distinguish different nodes and their connections ■ The visualization shall clearly distinguish properties of the graph through color coding, such as impact and interconnectedness	This will all be validated with visual confirmation tests
The product shall allow for the exploration of the graph	Interactive testing by the way of attempting to explore the graph
The product shall display the ability to search the graph in the user interface	Visual confirmation of the search dialog
The product shall allow for exploration of the graph via click or touch interface	Click/touch the visualization in various states
The product shall maintain an immutable search history easily accessible to the user for reuse of search terms	Attempt to change the search history, shouldn't be able to
The product shall display to the user when it is processing a user's search query	Visual confirmation test

Table 2: Test Plan

Project Schedule

Task	Start Date	Complete Date
Rough Specifications and Requirements	9/1/2015	10/28/2015
Prototype (three 2 week sprints)	10/2/2015	11/13/2015
Minimal Viable Product/Presentation	11/13/2015	12/4/2015
Requirements Finalized	12/4/2015	1/18/2016
Core Implementation (to be done in three 2 week sprints)	1/18/2016	3/4/2016
Implementation/Integration (1 sprint)	3/4/2016	3/18/2016
Testing Finished (1 sprint)	3/11/2016	3/25/2016
Final Implementation (three sprints)	3/18/2016	4/29/2016
Final Presentation/Delivery	3/25/2016	5/6/2016

Table 3: Tentative Project Deadlines

The prototyping phase this semester and the implementation phases will be developed in iterations, focusing on 2 week sprints. This will allow us to break down tasks and implement them more quickly.

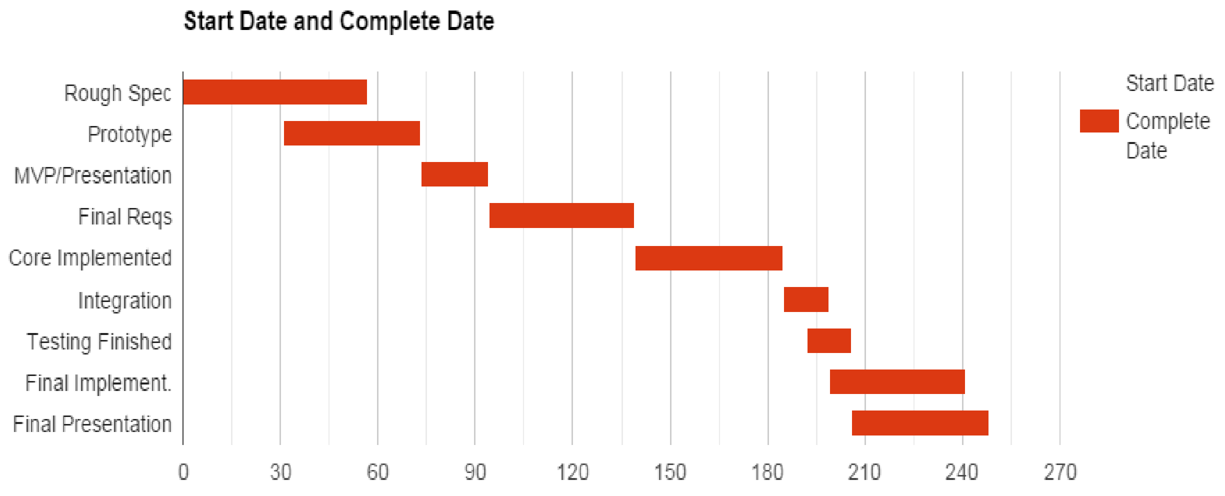

Figure B. Project Schedule

Appendix