May1618

Graphalyzer

A Graph Visualization and Analysis Tool

Team:

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

Advisor: Simanta Mitra

Client: Workiva

Project Plan v2.0

Table of Contents

1. 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, and performance may suffer as this grows larger. The goal of this project is to build a fast, generic graph visualization and analysis web application.

2. Similar Products

Graphalyzer is intended as a successor to an existing Workiva tool, so Graphalyzer's purpose is to provide the functionality of the previous tool, but circumvent the drawbacks from before. Workiva's previous tool used D3, a JavaScript library for visualizing document. One of Workiva's requirements is to be able to visualize and analyze **very large** graph files. The problem with the previous tool is that it was very slow, and lacked specific analysis features, and the technology stack involved was more built for data files as a whole, not just graph files. Graphalyzer is built for graph files only, and one of its main goals is to drastically improve performance for visualization and analysis.

3. Requirements

3.1. 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.

3.2. 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.

3.3. Backend Functional Requirements

3.3.1. The product shall allow for the upload of arbitrary graph files for analysis via REST

3.3.2. The product shall index incoming information associated with nodes and edges

3.3.3. The product shall allow for the searching of nodes by name

3.3.4. The product shall process the graph to determine:

- Depth
- Breadth
- Interconnectedness
- Impact

3.4. Frontend Functional Requirements

3.4.1. The product shall be able to visualize the graph such that:

- 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

3.4.2. The product shall allow for the exploration of the graph via click or touch

3.4.3. The product shall display the ability to search the graph in the user interface

3.4.4. The product shall maintain an immutable search history easily accessible to the user for reuse of search terms

3.4.5. The product shall display to the user when it is processing a user's search query

3.5. Backend Nonfunctional Requirements

3.5.1. The product shall be able to handle a large number of nodes (upper bound TBD) without crashing or hanging

3.5.2. The product shall support different file upload protocols (e.g. REST, FTP, etc.)

3.5.3. The product shall maintain original graph data integrity

3.6. Frontend Nonfunctional Requirements

3.6.1. The product shall be able to process, visualize parts of, and analyze graph files with sizes up to 32GB

3.6.2. The product shall not crash or lose connection to the REST service while in use

3.6.3. The product shall not have any memory leaks or any loss of graph data.

3.6.4. The product shall not voluntarily or involuntarily modify the graph data.

3.7. 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.

3.8. Interface Description

Graphalyzer's interface shall be a web browser client with three main components:

- Graph Panel that displays an interactive visualization of a graph, such that interacting with nodes of the graph prompts the interface to display more information
- Node Panel that describes information of a currently selected node from the graph
- Search Panel that allows the user to control what is displayed in the graph panel, such as the graph file, specific nodes, and size of breadth of connections from a node

4. Concept Sketch

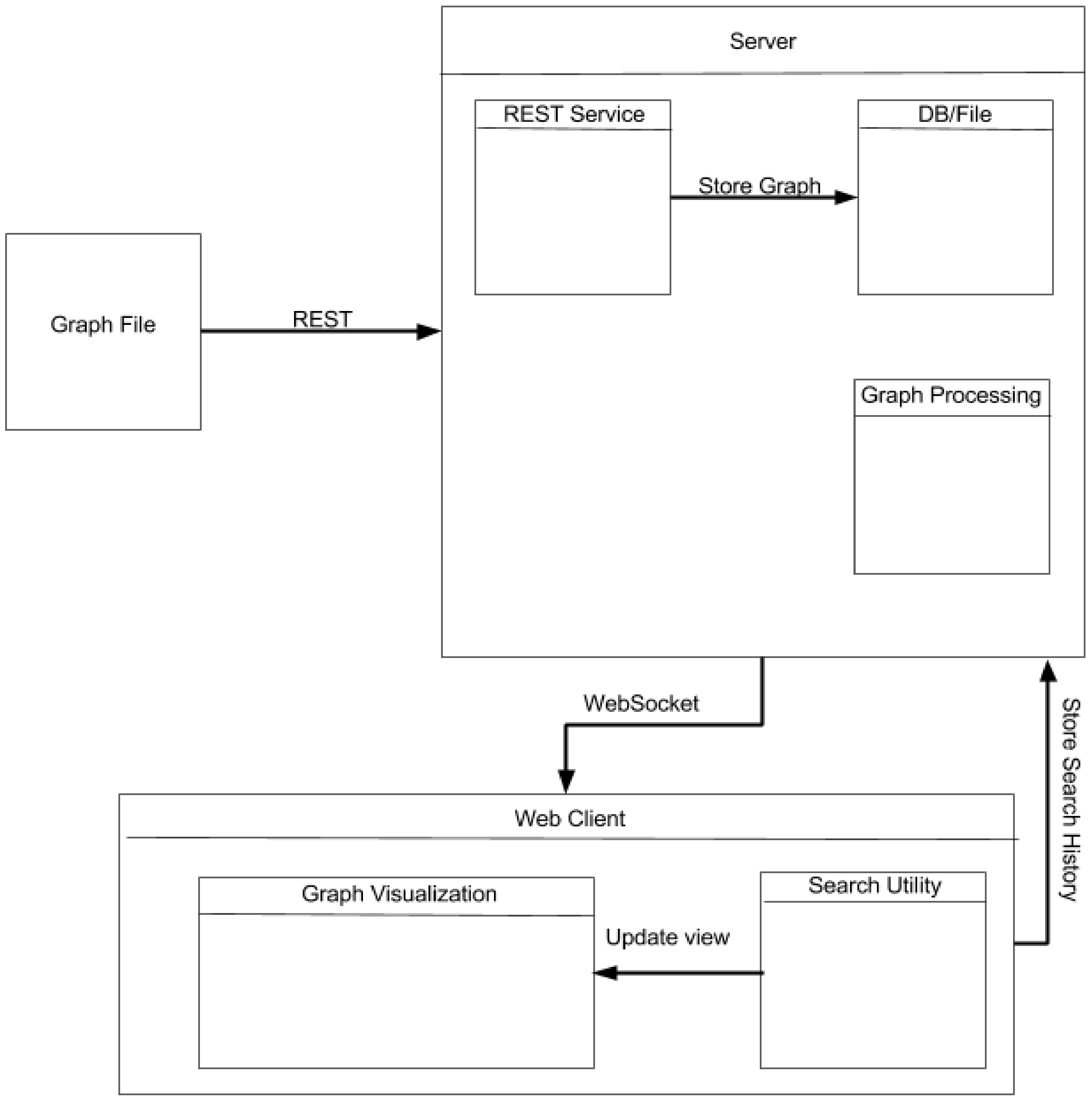


Figure A. Concept Sketch

5. Project Schedule

5.1. Work Breakdown

5.1.1. Sprints

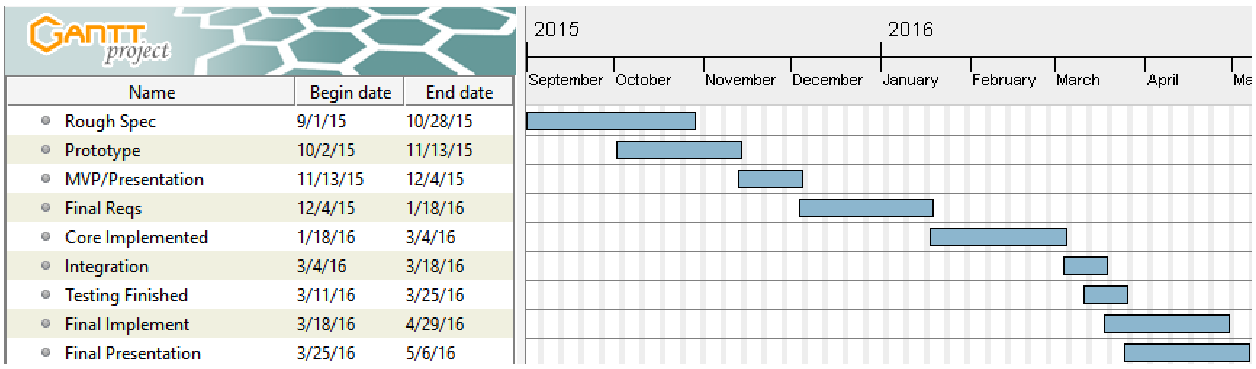
The prototyping phase this semester and the implementation phases will be developed in iterations, focusing on 2 week sprints beginning and ending on Fridays. This will allow us to break down tasks and implement them more quickly.

In one special case, a 3 week sprint will be used from 10/19/2015-11/6/2015, to line up the completion of the prototype by the end of Week 10 of the semester. This will allow the team to present a live demo of the first iteration of Graphalyzer to Workiva, and set up a launchpad for two more 2 week sprints before the 491 presentation in Week 14 and end of semester.

5.1.2. Work Management

We will be using Trello, an online project management tool with a wide spectrum of customization available. Trello acts as a sprint board. There are three main columns: **TODO, In Progress,** and **Done**. The board will populated with individual cards representing tasks that need to be done at the beginning of every sprint, and Trello will always be kept up to date on the status of each task. Each task is assigned to a person or group of people, and it is their responsibility to complete each task before its designated due date, which in most cases is the end of the sprint.

5.2. Gantt Chart



Name	Begin date	End date
Rough Spec	9/1/15	10/28/15
Prototype	10/2/15	11/13/15
MVP/Presentation	11/13/15	12/4/15
Final Reqs	12/4/15	1/18/16
Core Implemented	1/18/16	3/4/16
Integration	3/4/16	3/18/16
Testing Finished	3/11/16	3/25/16
Final Implement	3/18/16	4/29/16
Final Presentation	3/25/16	5/6/16

Figure B. Project Schedule

5.3. Meetings

5.3.1. Advisor Meetings

For the Fall semester, the team and Dr. Mitra will normally meet weekly, every Monday at 3:15pm. The purpose of these meetings is to maintain focus on short-term goals, as well as plan accordingly for long-term goals.

5.3.2. Client Meetings

For the Fall semester, the team and Workiva will normally meet weekly, every Friday at 3:30pm over Google Hangout. The main goals of these meetings is to provide status updates on project development, obtaining and clarifying requirements, and presenting demos of development iterations.

5.3.3. Team Meetings

The team will meet as needed, with a minimum being at least once a week. Generally, meetings will be conducted over Google Hangout after client meetings. The goal of these meetings is to keep everyone in the loop on progress with tasks in the currently active sprint. Additionally, time will be set aside in meetings at the end of each sprint to plan out the next sprint.

6. Development

6.2.1. Overview

We are using Git as our method of source control, with GitHub as a repository host. Two project repositories will be used; Graphalyzer and graphalyzer.github.io, with Graphalyzer being the official project repository, and graphalyzer.github.io being the project website.

6.2.2. Git Practices

All development on Graphalyzer will be done on feature branches specific to the feature being worked on - e.g. **'graph-panel'** for all development on the Graph Panel component on the frontend. When work is finished on a feature branch, a pull request will be opened with brief, but concise documentation on the changes represented. The pull request will be merged into the master branch only after a code review from all members of the time. Code reviews ensure that code is well-documented, tested, and consistent with the rest of the project. When every team member approves the changes, the owner of the pull request may merge.

7. Feasibility

Our team has extensive experience in various technologies, including different spectrums of web development such as REST, UI design, and JavaScript, and have all taken classes on graph theory. So over the course of 9 months, a project such as this should be feasible. See the next section for risks.

8. 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 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