Software Requirements Specification

1- Introduction

In today's world, open source software products are becoming increasingly popular to use in development over proprietary software. While this may in part be to avoid licensing fees there are many benefits to using open source software including no vendor lock in, high quality software and an abundance of support for this software. Although an arising dilemma with the huge increase in the use of open source software is that it with so many options it can be very difficult to decide which open source tool to use and also to see the quality of the community and contributors behind these projects. CHAOSS (Community Health Analytics Open Source Software) is a tool being developed to help with this new problem. It focuses on analyzing open source projects and creating visual metrics to measure the health of the community in these open source projects. Examples of these metrics include organizational diversity, code quality, value and risk. Our team aims to add new metrics to help CHAOSS effectively and accurately measure the health of these communities.

2 - Software Product Overview

CHAOSS is currently made up of three software products, Augur, Cregit and GrimoireLab of which we will focus on Augur but review all three. Augur could best be summarized as a Python library and REST server that provides data related to GitHub repositories. To go more in depth, all of the metrics provided in Augur are organized into the following categories evolution, experimental, risk, utility and value. The evolution category focuses on how the source code changes over time and how the project implements and controls these changes. It has metrics for commits, reviews and issues that are brought up. The experimental section is for metrics that are still in development. The goal of the risk category is to measure the community support and activity for a specific open source tool. This category measures things like issue resolution time, licensing and forking. The utility section provides useful information that may be used in any context. The value category aims to estimate the labor investment in these projects. It measures things like the number of starts and watchers in a given repository. The next software within CHAOSS is called Cregit and this is a framework of tools used to help facilitate the collection of metrics in git repositories. The last software tool in CHAOSS is called GrimoireLab and it is and open source tool that gathers data from many systems such as GitHub, Jira, Jenkins, StackOverflow and many more and then organizes this data into a database which can be integrated with many other tools.

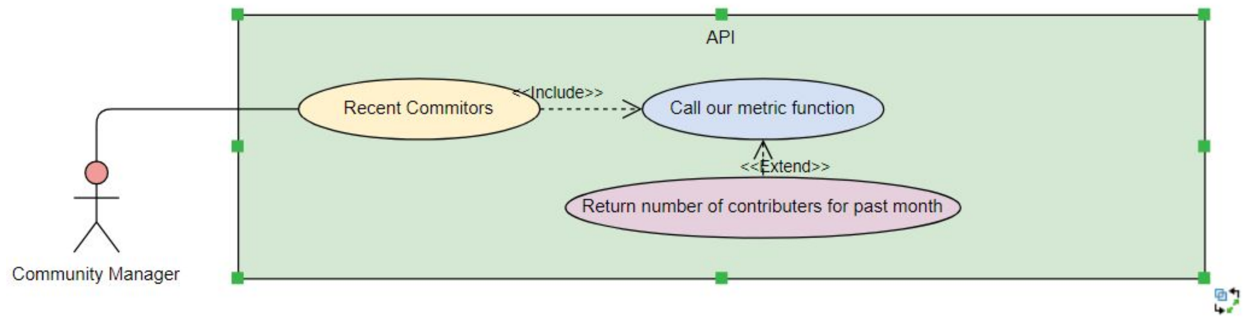
3 - System Use

Actor Survey

- Community Managers: Community Managers would be people in charge of managing and running the development/maintenance of an open source software product. CHAOSS would be a very useful tool for these community managers to help analyze how their open source project is doing and if the community support behind their project is declining or increasing. Using this tool they could gather many different metrics about their project and then based on this information they could make changes to address any problems that they discovered after this analysis. This tool could also be used to help compare several different repositories if they manage several and could help identify which ones are falling behind.

- Company Managers: Company Managers would be any person at a company that makes decisions on what software products the company will use to develop software. This person or people would want to make a very informed decision about what software they will be investing in because this would be extremely costly decision to reverse. Therefore because the use of open source tools are so popular in today's world it would be very beneficial for this person or people to know if the community for a new software tool they are interested in is declining or if this project is instead gaining support. There many metrics within CHAOSS that could be beneficial in helping make decisions about which open source products to use. CHAOSS could help save a company a significant amount of money if it prevents them from investing in an open source tool that has lost community support.

4 - System Requirements



Recent Contributors Use Case:
This function will be passed a repo and then using SQL function it will find the number of contributors that have made a commit in the past month. It will then add all of these numbers together and return the number of contributors that have been recently active in the project.

Purpose

The purpose of this endpoint is to show how many different people are still actively making commits into this directory and can indicate if there is little or a lot of support for this open source project. This will indicate which projects are being worked on more than others in a recent time frame.

Metric Functionality

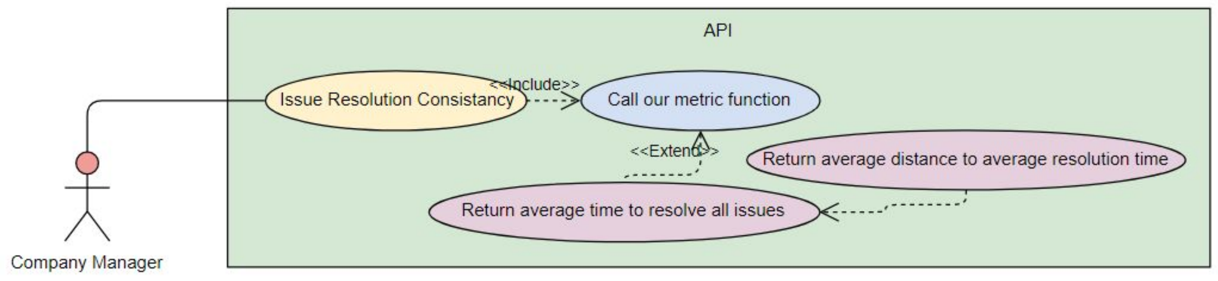
This function operates by first going to the data for the selected repo. It will then start adding to a counter every time a unique contribution made a commit in the past 30 days. It will then return this number to the API endpoint.

Tables Used

The tables needed for this endpoint are: commits and contributors

Non Functional Requirements

- Response Time
- Accuracy of measuring repository learning difficulty
- Metric consistency across different repositories


API
Issue Resolution Consistancy
<<Include>>
Call our metric function
<<Extend>>
Return average distance to average resolution time
Return average time to resolve all issues
Company Manager

Issue Resolution Consistency Use Case:
This function will be passed a repo and then using SQL function it will find the start and end timestamps for all issues within a repository. It will then find the difference between these times and find an average. Then using this average it will find the average distance away from this average number to see how consistently issues are resolved.

Purpose

The purpose of this endpoint is to show the variance of issue resolution times. This will let a company manager know if issues are being fixed at a consistent time or if the time it takes to resolve issues is scattered. If issues within an open source project are being resolved at a very consistent rate it may indicate that the repository is being managed very well.

Metric Functionality

This function operates by first going to the data for the selected repo. It will then find the average time for issues to be resolved and then using this number find the average difference between issue resolution time and the average resolution times. This value will then be returned to the api endpoint.

Tables Used

The tables needed for this endpoint are: repo and issues

Non Functional Requirements

- Response Time
- Accuracy of measuring repository issue consistency
- Metric consistency across different repositories

5 - Design Constraints

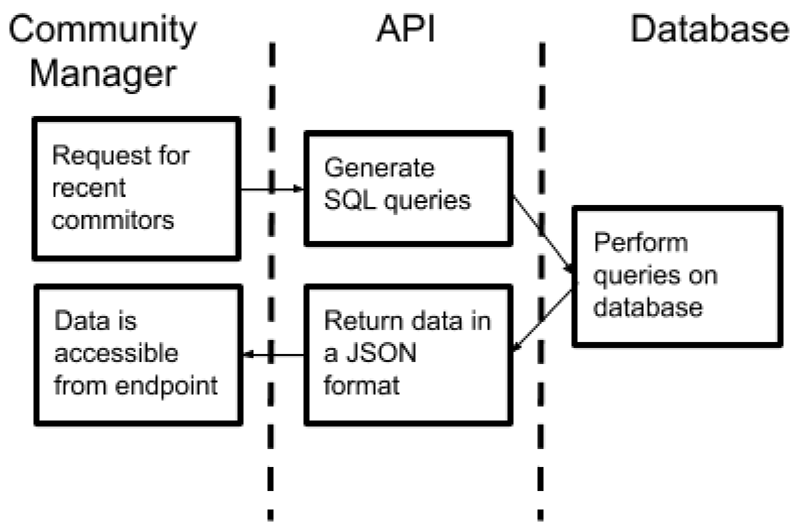
The following is a list of constraints that must be met in order to make the additions to CHAOSS outlined in the previous section

- Data: To add new metrics to CHAOSS there must be data from the repositories that is stored somewhere. The data that is measured must have information on contributors, issues, commits etc.
- Server: We would need to use a server to display and serve up the data that is requested from our API endpoints. This would be the location of where our code is running.
- Repositories: For these metrics to be accurate representations of the repositories we must have repositories that have been established for a decent time period and that people would be interested in having meaningful data analysis on.
- Database Accessibility: For these metrics to quickly and accurately gather data we must have access to a database that has stored information and statistics for all repositories that are currently being watched.
- Access to Endpoint: We must have an open endpoint on the running server that would allow people to access our endpoint and also allow us to return the data and information that they requested.

6 - Purchased Components

The need for possible purchased components will depend on the number of repositories that are currently being used for data analytics and also the amount of data that each one of these repositories needs. If the size is great enough there is a potential need for more data storage and also more bandwidth if the number of api calls is great enough. Without these purchased components the storage space may not be enough to hold all information about the repositories and the response time may become much slower if the number of api calls is increased.

7 - Interfaces


Community Manager
API
Database
Request for recent commitors
Generate SQL queries
Perform queries on database
Return data in a JSON format
Data is accessible from endpoint