Semester Project Sprint One

Andrew Cox, Kyle Malisos, Ali Dowlatshahi, Max Balk

Contribution Goals:

Purpose

The CHAOSS working groups have outlined a number of questions about open source community health and activity. Augur's answers to these questions are available in the form of GitHub api metrics. The number of the working groups' proposed metrics exceeds the number of metrics that have been implemented in Augur. Our contribution aims to provide implementations for three metrics to further the goals of the CHAOSS organization and improve Augur's insight-to-data ratio.

Use Cases

The CHAOSS project working groups have already done a lot of valuable research into community health and question development. In order to not waste time and effort potentially researching the wrong questions, we will be pursuing metrics that have already been proposed by CHAOSS project members. The three metrics our team will be creating are:

- **Top external organizations**: contributions by contributors whose organization is not the same as the software package's organization. Show top N external organizations for a given group of repositories.
- **Entrance difficulty**: Average time between an individual joining a community and their first contribution
- **Organizational distribution**: Contribution distribution by organization

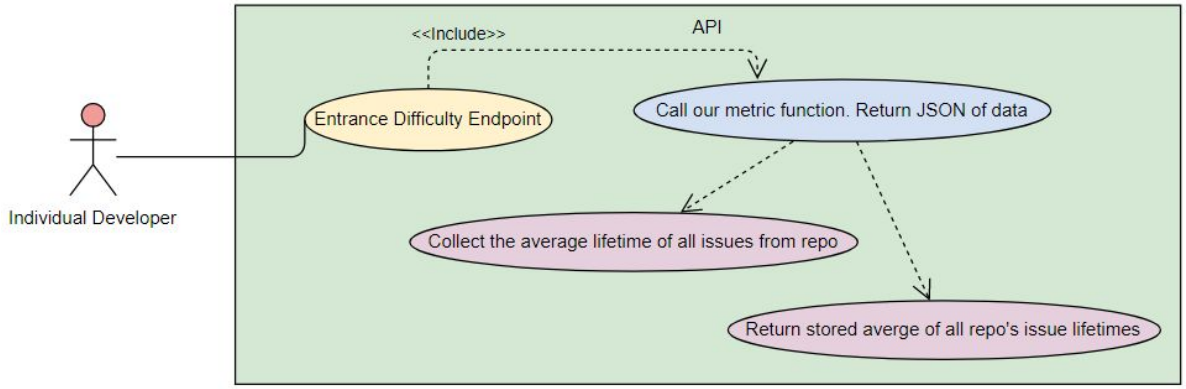
Software Overview:

Components

Each of our group's metrics will require additions to the relevant business object's main file and its routes file in the augur/metrics directory. Metric functions are defined in the business object's file and specify function parameters and SQL query, and perform any additional calculations before returning a pandas dataframe. In the routes file, we will use the server class's addRepoMetric or addRepoGroupMetric methods to expose our endpoints through the API and to the Augur frontend. When the endpoints are created on server startup, the server and application classes are made aware of which metric objects and functions to use when the server receives a request for a given endpoint.

Our architectural decisions are made to follow the rest of Augur's API layer structure. All metric functions seem to be created by tailored SQL queries that handle all of the data needs for each metric, so metrics are not coupled in any way. We assume this structure is used for speed because the DBMS is faster than the API layer.

Entrance Difficulty



The Entrance Difficulty metric will measure the average time between a fork of a repository and the first commit for this fork for all forks of the repository. This is designed to help measure how long on average it takes new members of a repository to understand the code enough to complete their first commit. This number will be returned with the average time across all repos in a repo group for comparison purposes.

Metric Functionality

The function operates by first getting the timestamp of when a each contributor's first forked the repository. It will then find the timestamp of when they created their first commit and find the difference between these numbers. These numbers will be retrieved through SQL queries.After getting the sum of this number for each contributor it will divide it by the number of contributors to get the average for the repository and it will return this value.

Relevant Data Tables: commits, contributors, repo

Top external organizations

Description

The top external organizations metric is related to the elephant factor, code quality, and skill demand metrics from both the risk and value working groups. We think a representation of which external organizations make the greatest number of contributions to a group of projects will provide insight to that project's health. This metric was first proposed in a November 2018 article by Sean Goggins.

Architecture

Endpoint

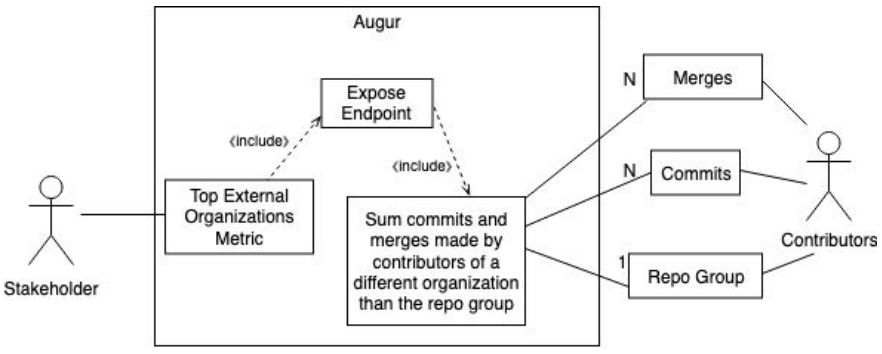
This metric will exist as a metric function and metric endpoint within the Augur application. Due to sample size and the nature of this metric as a summarized comparison, we plan to only make top external organizations available at the repo group level.

- A server.addRepoGroupMetric() function will be added to the Contributor routes file to expose our endpoint.

Metric Function

The contributions included in this metric will only be commits and merges. In the source article, it was proposed that comments would be included as well. Unfortunately, comments are messages and merge comments must be differentiated from issue comments, so we will be excluding those for now.
This metric can be handled entirely by its SQL query, so there will not be any further data manipulation in the metric function.
Our metric function will take in a self and repo_group_id parameter. Parameters relevant to timeseries data are not dealt with. Our resultant dataframe will include the repo group, organization, number of commits, and number of merges. The function's SQL query will gather all commits and merges that are done on repositories that are part of a repo group. Each commit or merge whose author belongs to an organization that is different from the repository's organization is summed for the contributor's organization. The query will be ordered by commit count, grouped by organization, and limited to the top five results.

Relevant Data Tables: commits, contributors, pull_requests, repo, repo_group



Organizational Distribution

Description

Organizational distribution is a metric that represents the ratio and number of contributions made by all of a project's contributing organizations, including the organization that governs the project. This metric is closely related to elephant factor and thus falls under the risk working group in the business risk focus area.

Architecture

Endpoint

This metric will exist as a metric function and metric endpoint within the Augur application. Because the data will be relevant at both levels, our endpoint will be available for both individual repositories and repo groups.

- server.addRepoGroupMetric() and server.addRepoMetric() functions will be added to the Contributor routes file to expose our endpoint.

Metric Function

The function operates by using the repo_group_id as a parameter to get all organizations that have worked on the project and the number of their commits. Then the total number of commits on a project are added from the organizations' commits and returned are each organization name, their number of commits and the ratio of their commits to the total commits made on that project.

Relevant Data Tables

commits, contributors