Semester Project Design Document

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:

- **Issue response time:** average time between an issue being created and the issue's first comment.
- **Organizational distribution**: Contribution distribution by organization
- **Issues messages over time:** Amount of messages on issues each week

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.

Issue Messages Over Time

Description

The Issue Messages Over Time metric will look at the amount of messages on issues in each week since its beginning. This will allow users to view the amount of activity and communication on issues over time to make it more apparent what repositories are being worked on and how well they communicate.

Architecture

Endpoint

This metric will exist as a metric function and metric endpoint within the Augur application. This metric will be available as both a repo and repo group metric. The repo group version will show the number of issue messages over all the repos in the group.

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

Metric Function

The function operates by first getting the issue messages that were connected to the repo or repos of the repo group. The issue messages will be grouped by the week they were created and the total number of messages in that week will be displayed. The function returns the week and the number of messages of that week.

Relevant Data Tables: issues, repo, message, issue_message_ref

Issue Response Time

Description

The issue response time metric is common to the evolution, risk, and perhaps the D&I working groups. This repository-only metric returns a time series of the average time between an issue's creation time and the time of its first comment. We think a visualization using this metric will help provide a measure of community communication health by showing trends and the ability to compare the current value against these trends.

Architecture

Endpoint

This metric will exist as a metric function and metric endpoint within the Augur application. This metric will be available as both a repo and repo group metric. The repo group version would show the average response time for any issues for repos in the group.

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

Metric Function

The two important pieces of data used for this metric are issue creation time and message creation time. For each issue in each repo, we will select the smallest difference between the issue and the timestamps of its messages, and then find a mean from each repo's issues. Comments (messages) are related to the issue table via a relational "issue message ref" table, so we will join message ref and then join message to pair each issue with its messages.

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, period and repo_id parameter. Since the response time for all issues in a repo are averaged, our resultant dataframe will consist of a series of average response times for each period.

Relevant Data Tables: repo, issue, message, issue_message_ref

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