Sprint 3 Design Document

EMMA BENNETT, STELLA CRAIG, ANDERS RYAN, BASHEER ABBAS SHAIK

Table of Contents

1 Scope
Summarized below are the design and implementation details for the two metric models chosen for this effort. Each group member implemented one metric per model. The division between what has been implemented in Sprint 2 in comparison to Sprint 3 is as follows:

Sprint 2 – Implementation of the Starter Health Model (due 12/3)

- Bus Factor – Stella Craig
- Time to First Response – Emma Bennett
- Release Frequency - Anders Ryan

Sprint 3 – Implementation of the Collaboration Development Index Model (due (12/7)

- Contributions – Anders Ryan
- Code Lines Changed – Emma Bennett
- Technical Forks - Stella Craig

Please see Stella Craig's branch of GitHub for up-to-date code demonstrating these metrics.

2 Introduction
Open-source projects have been critical throughout time in the development of unique, efficient solutions to a wide-ranging spectrum of computing problems. Broadly speaking, open-source software is software in which the copyright licensure allows users to modify or contribute to the underlying code in collaboration with other users and developers. Users can adapt software to personal needs, collaborate to produce updates, fix bugs, etc.

The health and sustainability of these open-source projects requires analyzing a multitude of different contribution statistics for a project over time. The ability to use these statistics to make meaningful conclusions about the project is hugely invaluable. Having current knowledge of a project's health and the ability to predict how contribution patterns will grow or change allow companies and organizations to concentrate resources, allows contributors to see the big picture results of their efforts, and provides tools for attracting new members and promoting the project.

The organization Community Health Analytics in Open-Source Software (CHAOSS) specifically develops useful health metrics and health metric models to allow for the assessment of the health of open-source projects and as well as provides a community for developers interested in advancing health metric tracking.

3 Metrics Overview
Selected for development in this project are the addition of two metric models to the existing 8Knot system. These models, as described below, will track useful project metrics which provide quantifiable feedback about project health and stability to the key actors that interact with this system.

3.1 Starter Project Health Metrics Model

The first metric model selected for addition to this system is the Starter Project Health Metrics Model. The information that builds this model, per CHAOSS, provides a simplified version of the most critical components related to project health. This model is intended to be a starting point, the first step an open-source project can take in furthering their understanding of the risk points and overall health of their project. These models are simple to understand, and generally applicable to most open-source projects. They include bus factor, time to first response, change request closure ratio, and release frequency (Metrics and Metric Models, 2023). These metrics are described in detail below.

3.1.1 Bus Factor

The bus factor metric represents the number of people that could feasibly leave a project without dropping the project contribution rate significantly. Bus factor is based on the number of contributors that make up 50% of contributions. This is a critical metric for project health and of particular interest to project stakeholders, as if a project is being carried by a single, or small amount, of contributors, it is at high risk for failure if those individuals stop contributing.

3.1.2 Time to First Response

Time to first response is a metric which tracks the average time a contributor's pull request sits before receiving interaction from a real-person maintainer for the project. The longer on average it takes for these requests to receive responses, the more at risk a project becomes as contributors can become discouraged if feedback time is slow. A general target for most projects is to stay within a ~2-day response window. This metric is in particular very helpful for project maintainers so that they can ensure adequate response speed.

3.1.3 Change Request Closure Ratio

Like time to first response, change request closure ratio is an important metric for project maintainers to track to make sure they don't become far behind on response times to their contributors. Change request closure ratio tracks how many total change requests have been merged or closed without merge. A higher deficit between total requests and closed requests indicates that the maintainers of the project are starting to lag, indicating a risk for the project.

3.1.4 Release Frequency

The goal of any open-source project is to release new updates and continue improving software. The frequency for these releases, anything from bug-fixes to new features, is an important metric to track to verify the project is seeing progress. Projects with low release rates are at risk for stagnation, and having a consistent release schedule indicates a healthy project. All releases, big and small, are accounted for in this metric. This metric also provides critical information to stakeholders, particularly those that may not be as technically fluent, about the robustness of how the project is progressing.

3.2 Collaboration Development Index Metrics Model

The goal of this metric model is to provide a quantifiable way to track the community environment surrounding an open-source project. Open-source projects are essentially representative of the collective intelligence of a group or community, and how that community works together and collaborates can make or break a project. The metrics within this model include contributors, downloads, code change commits, CI test, code commit linked with change request, ratio of issues to change requests, technical forks, and code change lines (Metrics and Metric Models, 2023). Only five, listed in this section, have been selected from this model for implementation as part of this effort. They are described in detail below.

3.2.1 Contributors

The contributors metric tracks the number of users (change requests, commit authors, reviewers) that have been active on the project within the past 90 days. This can be useful to track the ebbs and flows of a project and determine the overall trend. Projects which are trending downward in contributor activity may be at higher risk.

3.2.2 Code Change Commits

Code change commits track, per 90-day periods, the number of commits per week, the percentage of weeks with at least code commit, and the percentage of repositories with at least one commit (Metrics and Metric Models, 2023). This is a useful metric to analyze how active the coding community is on a project. Projects with waning activity are potentially at risk. This metric provides information like the contributors metric in that it seeks to quantify project activity, but it is more tailor focused on commits to the repositories themselves than the individual user interaction with the project (i.e. comments, reviews).

3.2.3 Change Request Reviews

This metric tracks the procedures for which change requests are reviewed and processed on a project. This can relate to the quality of review being completed on the change request (i.e. does the project have a formal contribution procedures document, if not do they need one). The metric provides analysis of how many change requests were reviewed, accepted, commented on and the number of contributors providing reviews versus authoring change requests over a 90-day period. The percentage of change requests who had at least one non-author reviewer are also tracked. This metric can be useful, for example if a project becomes bogged down with many rejected change requests. This puts a project at risk for stagnation as the deficit grows between when contributors submit a request and when maintainers will get to it, and it can also lead to maintainer burnout. Projects heavy in rejected requests may benefit from providing more instructional information to their users to communicate expectations for change request type.

3.2.4 Technical Forks

Technical forks are instances of a code repository that an individual copies over to their platform account to use and modify. Forking is an excellent tool which allows contributors to experiment with project code without effecting the main source. Forks can be contributing forks, or forks which regularly open change requests, or non-contributing. The technical forks metric quantifies how many copies of a project are in distribution at any certain time. This metric is useful to track as it can indicate how users are interacting with the code using platform tools. A high number of forks for example, while not necessarily a project risk, may result in unnecessary difficulties in merging later if major conflicting contributions are made on different contributing forks without communication.

3.2.5 Code Change Lines

Like contribution information, quantifying the amount of actual physical code edits is a great way to provide a general volume of project activity. Code change lines is a metric which tracks the number of lines (added and removed) that are changed over a specific period. This metric provides granularity that contributions does not as the number of lines changed can be large or small within a single commit.

4 Starter Health Metrics Design – Sprint 2

The following section describes further detail about the architectural and detailed design components of the Starter Health Metric Model's implementation in Sprint 2 of this effort. As this is an additional feature to a preexisting software with frontend and backend design, the focus of this section is describing how the backend database will be queried/accessed and how the existing fronted stack will be utilized to build intuitive visualizations which display each metric. The Starter Health metrics can be found in a designated tab of the 8Knot frontend.

4.1 Bus Factor - Stella

4.1.1 Database Querying

To get the appropriate content to construct the bus factor graph the materialized view "augur_data.explorer_contributor_actions" was queried for repository ID, repository name, contributor ID, created at, the login, action, and rank. This query was done through an additional queries python file which was added to the index callback function under the tag "ctq". The bus factor file was then used to generate the appropriate data frame.

4.1.2 Metric Visualization

The visualization chosen for this metric was a pie chart which displays the top contributors by commit. This was selected as it provides a clear visual of the minimum number of contributors that made up a certain percentage of commits. The contributors shown are the contributors that made up the largest percentages of commits. The number of contributors on display can also be edited through a text box below the graph. Additionally, for each pie slice the contributor id and number of contributions will be displayed to the exact number if hovered over.

4.2 Time to First Response - Emma

4.2.1 Database Querying

To get the appropriate content to construct the time to first response graph the materialized view "explorer_pr_response_times" was queried for repository ID, repository name, hours to first response, and pull request closed. A check was added to ensure only pull requests that had been closed and did not have a null time to first response were added to the data frame used to generate the graph. This query was done through an additional queries python file which was added to the index callback function under the tag "rtq". The time to first response visualization file was then used to generate the appropriate data frame.

4.2.2 Metric Visualization

The visualization chosen for this metric was a bar graph which displays, in mean hours, the time to first response for an individual pull request for a defined repository. This was selected as it provides a clear visual of any dates which the mean response time was particularly high, it also allows for multiple dates to be compared over a specified time range. As clarified in the metric description, the target response time is 48 hours (or 2 days). Pull requests are organized by the date on which they were closed. Mean first response times can be organized by day, week, month, and year through interactive fields. The date range on display can also be edited through a draggable frame below the graph. Additionally, for each bar the date and mean response time in hours will be displayed to the exact number if hovered over.

4.3 Release Frequency - Anders

4.3.1 Database Querying

To retrieve the correct data for the visualization the "releases" table was queried for repo_id, release_id and release_published_at by a selected repo_id. This data was then used to count the total number of releases over a certain period. This query was created in a separate release_frequecny_query.py file and then used in the release_frequency.py file to display the queried data in a data frame as a bar graph.

4.3.2 Metric Visualization

The visualization for the release frequency was chosen to be a bar graph that shows the total number of releases in a certain time interval selected by the user. The user can either select the time intervals of day, week, month to year for the data to be shown over. A bar graph was selected to visualize the data because it provides a clear and effective way to display the number of releases over certain periods of time. The user is also given the ability to select the total date range of the data via a draggable frame below the bar graph. This allows the user to see how the number of releases has changed over a longer period of time. When each bar of the bar graph is hovered over with the mouse it will display the exact period (day, week, month year) and number of releases that bar represents.

5 Collaboration Development Index Model – Sprint 3

The following section describes further detail about the architectural and detailed design components of the Collaboration Development Index Model's implementation in Sprint 3 of this effort. As this is an additional feature to a preexisting software with frontend and backend design, the focus of this section is describing how the backend database will be queried/accessed and how the existing fronted stack will be utilized to build intuitive visualizations which display each metric. The Collaboration Development Index metrics can be found in a designated tab of the 8Knot frontend.

5.1 Code Change Lines - Emma

5.1.1 Database Querying

To get the data regarding the lines added and removed per pull request "explorer_pr_response_times" was queried using the response time query (rtq) created for the time to first response graph in Sprint 2. Queries for lines added, lines removed, and total lines changed were added to the existing query, and a check was added to ensure only pull requests that were closed and had more than null changed lines were included in the pulled data. The code change lines visualization file was then used to generate the appropriate data frame and graph.

5.1.2 Metric Visualization

The visualization chosen for this metric was a bar graph which displays total changed lines per pull request, organized by pull request closed date and selected repository. Total changed lines encompasses the summation of added and removed lines. A bar graph was selected as it provides a clear visual of any dates which the number of changed code lines was particularly high, it also allows for multiple dates to be compared over a specified time range. This allows us particularly to see dates where large changes occurred (indicating a significant change in the project). The total code lines changed can be organized by day, week, month, and year through interactive fields. The date range on display can also be edited through a draggable frame below the graph. Additionally, for each bar the date and the total lines of code changed per that date or date range will be displayed to the exact number if hovered over.

5.2 Contributors - Anders

5.2.1 Database Querying

To get the number of contributors at a certain time I used the already existing contributors_query.py file which queries the explore_contributor_actions visualization in the augur_data database. This query returns the necessary data for this visualization of the date a contribution is made and who made the contribution

5.2.2 Metric Visualization

This visualization was chosen to be a line graph that shows the total number of contributors per period of time being weeks, months or years on the y axis and the date on the x axis. A line graph was chosen for this visualization because it is a clear way to see the way the number of contributors on a project increases and decreases over time. This is a useful metric because it allows the ebbs and flows of a project to be clearly seen indicating when a lot of people are working on the project at once. The total date range of the graph can be changed via an interactive frame below the graph. The graph can also be changed by 3 selectable options below the graph of week, month and year which when selected change the period slice contributors are counted over.

5.3 Technical Forks – Stella

5.3.1 Database Querying

To get the number of forks at a certain time I created and used the forks_query.py file which queries the repo table in the augur_data database. This query returns the necessary data for this visualization of the date a repo is made and where that repo is forked from.

5.3.2 Metric Visualization

This visualization was chosen to be a line graph that shows the total number of forks per period of time being weeks, months or years on the y axis and the date on the x axis. A line graph was chosen for this visualization because it is a clear way to see the way the number of forks on a project increases over time. The total date range of the graph can be changed via an interactive frame below the graph. The graph can also be changed by 3 selectable options below the graph of week, month and year which when selected change the period slice contributors are counted over.

6 Works Cited

(2023, Sept 18). Retrieved from Augur Documentation: https://oss-augur.readthedocs.io/en/main/index.html

(2023, Sept 18). Retrieved from chaoss / augur: https://github.com/chaoss/augur

Metrics and Metric Models. (2023). Retrieved from CHAOSS: https://chaoss.community/kb-metrics-and-metrics-models/