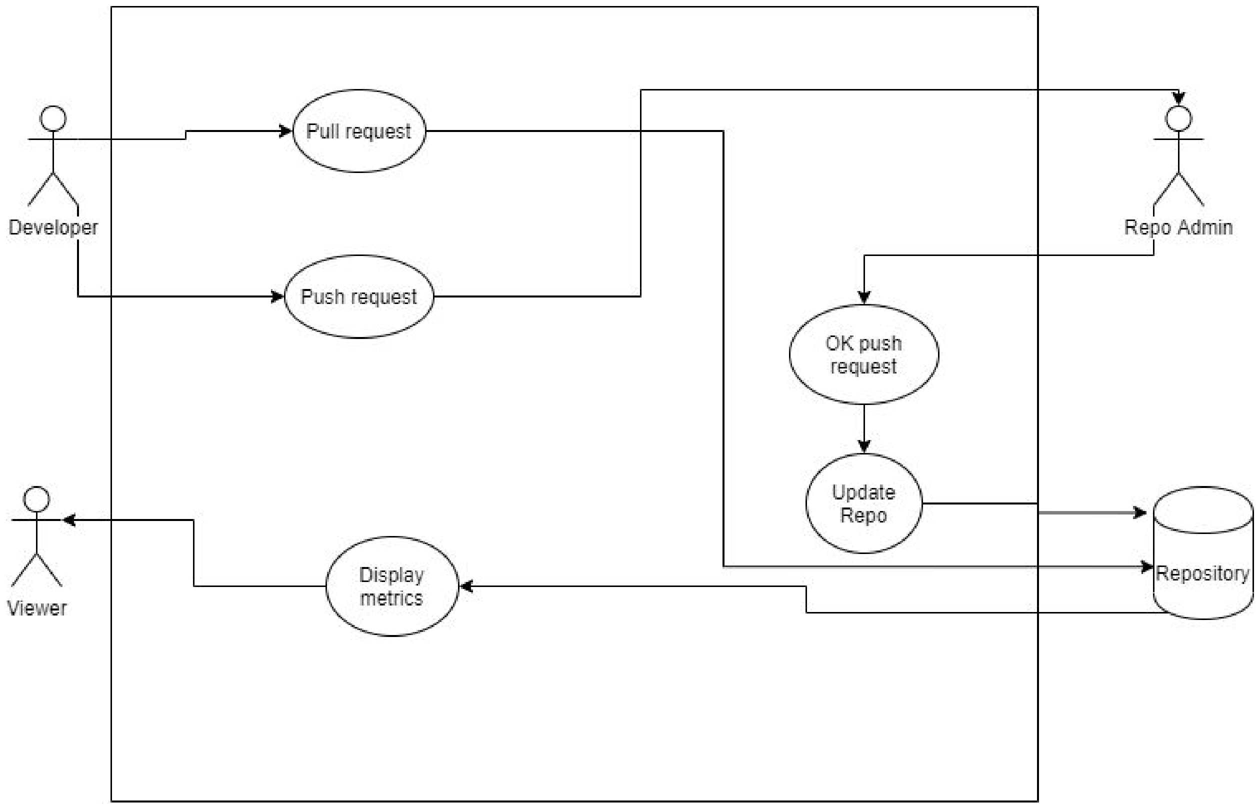
# Software Requirements Analysis

Jeremy Schneiderjohn

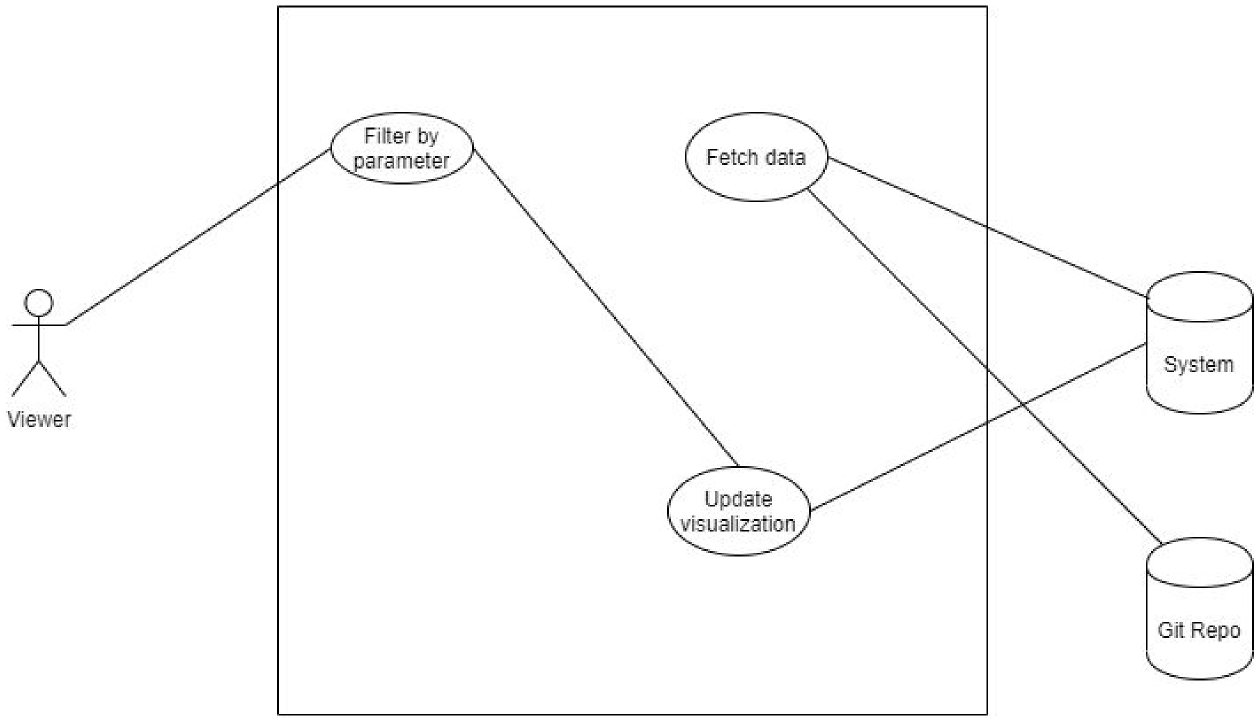
Given metrics description: Released metrics are only a subset of metric ideas that are being developed. If you would like to learn more and discuss different metrics please visit the working group repositories. The metrics are sorted into Focus Areas. CHAOSS uses a Goal-Question-Metric format to present metrics. Individual metrics are released based on identified goals and questions. The metrics include a detail page with definitions, objectives, and examples.

What it does: The application keeps track of the changes made to each repository. It can display how many changes were made by each person, the size of the changes, the pull requests, and the open issues.

Actor Survey:



System Requirements:
Code Change Metrics:



Objectives: Volume of coding activity. Code changes are a proxy for the activity in a project. By counting the code changes in the set of repositories corresponding to a project, you can have an idea of the overall coding activity in that project. Of course, this metric is not the only one that should be used to track volume of coding activity.
Implementation:

1. Aggregators:
    a. Count. Total number of changes during the period.
2. Parameters:
    a. Period of time. Start and finish date of the period. Default: forever. Period during which changes are considered.
    b. Criteria of source code. Algorithm. Default: all files are source code. If focused on source code, criteria for deciding whether a file is a part of the source code or not.
3. Filters
    a. By actors (author, committer). Requires actor merging (merging ids corresponding to the same author).
    b. By groups of actors (employer, gender...). Requires actor grouping, and likely, actor merging.
    c. By tags (used in the message of the commits). Requires a structure for the message of commits. This tag can be used in an open-source project to communicate to every contributor if the commit is, for example, a fix for a bug or an improvement of a feature.
4. Visualizations

    a. Count per month over time
    b. Count per group over time

Tools Providing the Metric:

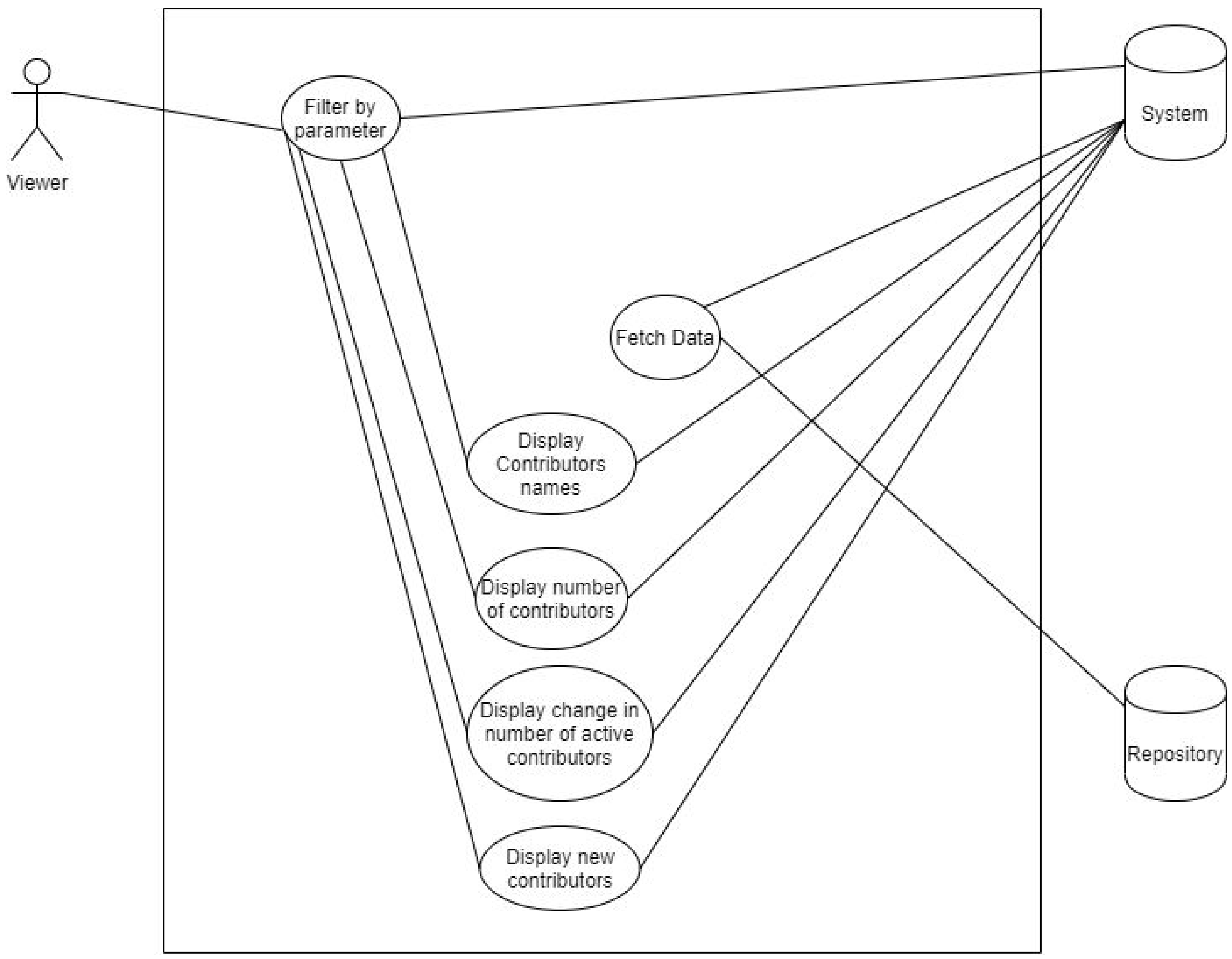
1. GrimoireLab metric box
2. Augur metrics
3. Gitdm

Data collection strategies:

Mandatory parameters (for Git):

1. Date type. Either author date or committer date. Default: author date.
   For each git commit, two dates are kept: when the commit was authored, and when it was committed to the repository. For deciding on the period, one of them has to be selected.
2. Include merge commits. Boolean. Default: True.
   Merge commits are those which merge a branch, and in some cases are not considered as reflecting a coding activity.
3. Include empty commits. Boolean. Default: True.
   Empty commits are those which do not touch files, and in some cases are not considered as reflecting a coding activity.

Contributors Metric:


Viewer
Filter by parameter
System
Fetch Data
Display Contributors names
Display number of contributors
Display change in number of active contributors
Display new contributors
Repository

Description:  A contributor is defined as anyone who contributes to the project in any way. This metric ensures that all types of contributions are fully recognized in the project.

Objectives:  Open source projects are composed of a number of different contributors. Recognizing all contributors to a project is important in knowing who is helping with such activities as code development, event planning, and marketing efforts.
Implementation:  Collect author names from collaboration tools a project uses.
Aggregators:

- Count. Total number of contributors during a given time period.

Parameters:

- Period of time. Start and finish date of the period. Default: forever. Period during which contributions are counted.

Filters:
By location of engagement. For example:

- Repository authors
- Issue authors

- Code review participants
- Mailing list authors
- Event participants
- IRC authors
- Blog authors
- By release cycle
- Timeframe of activity in the project
- Programming languages of the project
- Role or function in project

Visualizations

1. List of contributor names (often with information about their level of engagement).
2. Summary number of contributors.
3. Change in the number of active contributors over time.
4. New contributors (sort list of contributors by date of first contribution).

Tools providing metric:

1. GrimoireLab
2. Augur

Data Collection Strategies:

As indicated above, some contributor information is available via software such as GrimoireLab and Augur. However, some contributor insights are less easily obtained via trace data. In these cases, surveys with community members or event registrations can provide the desired information. Sample questions include:

- Interview question: Which contributors do not typically appear in lists of contributors?
- Interview question: Which contributors are often overlooked as important contributors because their contributions are more "behind the scenes"?
- Interview question: What other community members do you regularly work with?

Additionally, surveys with community members can provide insight to learn more about contributions to the project. Sample questions include:

- Likert scale [1-x] item: I am contributing to the project
- Matrix survey item: How often do you engage in the following activities in the project?
    - Column headings: Never, Rarely(less than once a month), Sometimes (more than once a month), Often(once a week or more)
    - Rows include: a) Contributing/reviewing code, b) Creating or maintaining documentation, c) Translating documentation, d) Participating in decision making about the project's development, e) Serving as a community organizer, f) Mentoring other contributors, g) Attending events in person, h) Participating through school or university computing programs, i) Participating through a program like Outreachy, Google Summer of Code, etc., j) Helping with the ASF operations (e.g., board meetings or fundraising)

Functional Requirements:

1. Must display Correct data
2. Data must be accurate.
3. Must keep track of all parameters.
    a. Period of time: Start and finish date of the period. Default: forever.
       Period during which changes are considered.
    b. Criteria for source code; Algorithm Default: all files are source code.
       If we are focused on source code, we need a criterion for deciding whether a file is a part of the source code or not.
    c. Type of source code change:
        i. Lines added
        ii. Lines removed
        iii. Whitespace

Non-functional Requirements:

1. Shall display correct data for the corresponding time frame.
2. Site shall have a good connection to the database.
3. Shall be able to display metrics by filter.

Design constraints:

1. All non-functional and functional requirements.
2. Works with Augur, GrimoireLab, andGitdb.
3. Large enough hardware to accommodate the site.
4. Built with python, flask, and REST server.
5. Open source, must manage push requests.

Purchased components:

1. Server to host website
2. ISP to give internet to server.
3. Domain name.

Interface:

1. The CHAOSS website.