Name: Deepayan Ghosh
Email: deepayanghosh89@gmail.com
Phone: +91-7980132889
GitHub: https://github.com/Deepayan-Ghosh

Reporting CHAOSS Metrics using Manuscripts

Synopsis

Currently, Manuscripts produces PDF report based only on following sections: ”Overview”, ”Activity”, ”Community” and ”Process”. However this is not enough for getting a comprehensive overview of a project. Our target for GSoC is to enhance *Manuscripts* so that it produces reports based on CHAOSS metrics. First part of my work involves writing Python code to retrieve relevant information for the metrics, producing suitable representation for them and integrating them into Latex, which will make Manuscripts, a full-fledged reporting tool based on CHAOSS metrics. Second part involves adapting the current Grimoirelab tool-chain to produce similar reports from raw data.

Benefits

Our work on Manuscripts will make it a tool which produces generic reports based on CHAOSS metrics. It will allow the owners of a project and the community in general, to assess risks involved with the project and its growth, in terms of code, value, community-diversity and community-health from a single comprehensive source. Since *“A picture is worth a thousand words”*, proper visualization of the data will allow community people to understand conditions of the project better.

Deliverables

Project Goals

- First analyze every metric and write down the data which is required for its calculation and representation. Also study GitHub API and GitHub Search API.
- Next, obtain these data. Some data is already available from elasticsearch indices, some needs to be obtained manually while others are available through APIs. However, methods which extract such data from APIs doesn’t exist in *Perceval* and hence, needs to be implemented.
- Decide proper representation such as bar chart, pie chart, line chart,etc. for each metric.
- Implement the above in Python code, write tests and add documentation.

- **Mordred** currently produces report in two steps: produce index, then call *Report.py*. We will bypass indexing stage, parse raw-data, and produce reports. Details below.

Proposed Implementation

Enhancing Manuscripts: Some metrics can be obtained using Github API and git backends directly but the code for them do not exist in Perceval. For example, number of lines in source code can be obtained as given here, licence information can be obtained from GitHub API licences, number of downloads (here), number of forks (here), and many more. However such methods do not exist and will be added to Perceval.
Other metrics cannot be obtained using API, and have to obtained manually. We will do this using external files (JSON, YAML or CSV format), and obtain data by parsing them. I have done a crude sample implementation here. There will be one master JSON file which will contain paths to all the external files listed according to metric category. For example, if category "diversity_inclusion" needs file1 and file2 then the master JSON file should contain {"diversity_inclusion": ["path-to-file1", "path-to-file2"]}.The path of the master JSON file will be accepted as command args in manuscripts, then it will be read and passed on to report.create().
We will add section methods similar to sec_overview() for each metric, with file-paths needed by that metric passed as arguments. We then modify the files under metrics, for example, class method get_section_metrics() inside metrics/git.py should return the new metric categories along with their sub-categories properly listed. Also, add the sub-categorical activity-metrics as classes which inherit Metrics class, and enlist them in activity_metric list so that they can be called from here. Inside *Metrics* class, we modify *get_query()* accordingly to produce required new query strings and add new methods required for calculating the metrics. Finally, modify *secs* in sections() to contain mappings for *sec_diversity_inclusion(files)*, *sec_growth_maturity-decline(files)* etc.
New visualizations: To implement the new visualizations we implement new methods if needed, and call these from __create_csv_eps() method. For example, if we agree on *line-charts* or *pie-charts*, then we need to add new methods which will follow the same format as bar3_chart() and bar_chart(). These will be made using *matplotlib* and *prettyplot* libraries.
Adapting Mordred: The aim is to obtain the raw data from Perceval, bypass index creation in Elk, call *report.create()* with raw-data, parse it and produce the report using Manuscripts.The class *Task_Report* is at the end of all_task list in *Mordred* class. So before creating report, indices are created. Currently Mordred uses *Elk* in this line to produce the indices. Following the function-calls through Mordred to *Elk*, I found that inside feed() we obtain the data from Perceval in items and then *feed_items()* accepts it to produce the indices. We will bypass this, and return *items* directly. Usually, it is inside Manuscripts that we retrieve data from indices and files. However, if report.create() is called from Mordred (indicated by a flag var) we bypass querying indices and

instead, parse the raw-data separately to retrieve required data, call the new section functions with parsed data which then calculates the metrics and produces PDF report. Files are parsed as before.

Related Work: I have completed two microtasks and sample implementation of three metrics here, which reads data from external sources as proposed above, and produces visualizations.

Proposed Timeline

- **Milestone 1 (April 24 - May 13):** Mainly work on points 1 and 2 of project goals. I tried to run Manuscripts with GitHub, it did not work. So we need to make sure all data sources are working. Besides this, study *Mordred* source code, fix bugs and issues to understand the source code better.
- **Milestone 2 (May 14 - Jun 12):** The target is to complete implementation of all CHAOSS metrics by 1st evaluation phase.
- **Milestone 3 (June 16 - July 11):** Complete writing code for the visualizations (bar charts, line charts, etc), document the code and write proper tests for the new features. Start working on Mordred.
- **Milestone 4 (July 14 - August 6):** Complete working on Mordred, write proper documented and working tests. If possible, assess producing Jupyter notebooks as well.

I will be able to devote 48 hours per week during May, June, and starting of July, and after that 40 hours per week.

Biographical Information

I am Deepayan Ghosh currently in 3rd year and pursuing B.Tech in Computer Science and Engineering at KIIT University. I have been working with Python for more than a year now. I can use Jupyter notebooks and have fair amount of experience with latex. I know *sqlite3* module but never used it in projects. However I did use JDBC for my projects so picking up database interfacing in Python will be easy.
I came across **Grimoirelab** in GSoC organizations list for the first time. **Manuscripts** interested me the most because an open-source reporting tool is like a software tool that can produce a diagnostic report of a patient (here, a repo) containing all the information in one place, which is free and can be used by anyone. Integrating CHAOSS metrics is a great idea as it covers all metrics necessary for proper reporting in different areas. I would consider myself fortunate to be able to be a part of this. Incorporating CHAOSS metrics will require a lot of work from scratch, and this will give a taste of how Grimoirelab toolchain works, and there can be nothing more exciting than working on a great project, from scratch, and getting to learn its internals.