Support of Source Code Related Metrics

Adding support to Grimoire Lab to produce source code related metrics using Graal

Abstract:

The **Graal Project** allows us to conduct a customizable and incremental analysis of source code by leveraging on existing tools and producing an output that conforms to the data that can be processed by GrimoireLab. Graal already offers analysis about code complexity, quality, dependencies, security and licensing, however, currently it is not integrated with GrimoireLab. The aim of this project is to add **Graal** to the GrimoireLab toolchain in order to produce source code related Metrics. *I will mainly be working on defining and calculating metrics related to source code analysis data provided by Graal, enhancing Graal to support more analysis and improve existing ones, producing analytics with the proposed and implemented metrics and including them in Sigils, adding unit tests and documentation related to work done.*

Benefits to Community:

Source code analysis tools play a key role in maintaining a software development project and is crucial for sustaining open-source development. The Graal Project will help provide the end-users, moderators with a better understanding of the **source code** and **software health** with the help of analysis provided. It would empower users with a customizable, scalable and incremental approach to conduct source code analysis and enables relating the obtained results with other software project data.

Goals:

Graal produces analysis related to code complexity, quality, dependencies, vulnerability & licensing and it conforms to the data that can be processed by GrimoireLab. I will mainly be focusing on:

1. Adding support of **source code related metrics** to Grimoirelab with the help of analysis data produced by Graal.
2. Adapting Grimoirelab toolchain to be able to execute **Graal** and process the data produced by it.
3. Writing appropriate **unit tests** for additional backends, their corresponding supporting connectors, and methods.
4. Producing analytics related to proposed and calculated **metrics*** (described below)
5. Adding **documentation** related to additional features and improvements in existing ones.

Out of all the five backends provided by Graal, **CoCom** (Code Complexity) covers a vast majority of the popular languages and **CoLic** (Code License) supported by NOMOS & ScanCode helps us fetch license & copyright related information from software development repositories and is language independent. Addition of metrics related to these two backends during GSoC period could be applied to a wide range of projects in the future.

***Note**: As per the current state of the Graal project, **CoQua** (Code Quality), **CoVuln** (Code Vulnerability) & **CoDep** (Code Dependency) backends do not support other popular languages except for Python, hence the implementation of metrics related to the data produced by these backends would require additional analyzers to be added to Graal.*

Metrics to be Implemented:

Performing license & copyright related analysis on a given git repository using **CoLic** backend with some enhancements can suffice to implement the following metrics:

1. **All licenses:** List of licenses present in the repository.
2. **File License Declarations:** A list of license declarations on the software package files.
3. **License Count:** Number of licenses found in a software repository.
4. **Package License Declaration:** A list of license declarations on the software package.
5. **Copyright Declaration:** The degree to which the project properly declares copyright details.
6. **License Coverage:** Number of files with a file notice (copyright notice + license notice) and ones without a file notice.

Performing code complexity analysis (cyclomatic complexity, LOC) on a git repository using **CoCom** backend (which leverages on CLOC & Lizard), provides information which can help define and calculate metrics such as:

1. **LOC:** Total Lines of Code
2. **CCN:** Cyclomatic Code Complexity of a software repository.
3. **Complex Files:** A list of files with higher complexity in terms of Cyclomatic Code Complexity (CCN) & Lines of Code (LOC).
4. **FUNS:** Number of functions
5. **Comments:** Total comment lines present in the repository.

***Note**: Some of the implementations might even require new analyzers to be added to Graal.*

(Feasibility and Priority of addition of metrics listed above will be decided after discussion with mentors and once approved, will be opening an issue ticket, interacting

with the Working Group, understand its context and procedures in order to move ahead with the work.)

Implementation Details:

Phase #1: Defining and Implementing Source code related metrics

1. This phase would involve having a discussion related to the above proposed metrics and incorporating changes to it, defining the discussed and finalized metrics in case they aren't already under **Metrics*** defined by the CHAOSS Metrics Committee, adding their implementation details via Issues and Pull requests to the corresponding repository.
2. Adding implementation of the defined and accepted metrics to be calculated and stored in the enriched indices. As the data produced by Graal backends is very much similar to the data which can be processed by **GrimoireLab** toolchain, work on adapting **ELK** and **Mordred** to be able to execute Graal and process the data produced.

Phase #2: Writing Tests, Producing analytics with Graal data & including them in Sigils

1. This phase would start with writing unit tests to cover the methods built in the earlier phase and writing necessary documentation for the methods created.
2. The main task to be done in this phase would be the creation of dashboards in **Kibiter** to produce suitable representation for the implemented metrics, such as tables and charts and include the panels in **grimoirelab-sigils**

Phase #3: Testing & Evaluating the implementation

1. This phase would involve picking projects of different sizes and benchmark the time needed by **Graal** to process them. This would also allow us to better understand in defining a scope of improvements in future by the enhancement effort in other backends to support analysis related to other popular languages and where optimization efforts should be applied in the future.
2. A buffer time suggestions & improvements in the implementation and in case of uncontrolled delay.

Timeline:

Until 6th May: Until the end application review period, I plan to discuss and work on improvements that can be made in the existing analyzers, the addition of new ones to the appropriate backends and understand the GrimoireLab toolchain better. The corresponding thread for enhancements/improvements in analyzers can be found **here**.

May 6 to May 27 (Community Bonding): This phase would involve understanding how the community works and how I can participate in the discussion that happens, setting-up a clear way of communication with the mentors, determining the tasks that are of highest urgency to complete, adding discussed metrics to Metrics repository, discuss alternative methods for calculating the Metrics, what metrics except for the existing ones are feasible to implement and within the scope of the project.

May 28 to June 23: Work on enhancing **grimoirelab-elk** so that the data produced by Graal can be stored in raw indices, the relevant metrics can be calculated with the help of analysis produced and can be stored in the enriched indices. The task would involve adding more **connectors** to grimoirelab-elk, each connector consisting support of the corresponding graal backend, **raw** & **enrich** connector and graal command as per the convention. Accordingly, modify **Grimoirelab Mordred**'s methods and configurations to accommodate changes made to grimoirelab-elk.

(The above defined task is expected to be completed before Phase-1 Evaluation Period)

June 24 to July 2: After enhancement of **grimoirelab-elk** (as described above) and supporting those changes on Mordred, write unit tests for the functions that are created till now, improved test coverage for added enhancements and add necessary documentation.

July 3 to July 22: Focus on creating tables and charts necessary to show source code related Metrics calculated and stored in enriched indexes. Creating dashboards and adding panel related JSON documents to ***grimoirelab-sigils***. Test loading newly created dashboards in **Sigils** via Mordred.

(The above defined task is expected to be completed before Phase-2 Evaluation Period)

July 23 to August 13: This period would involve benchmarking the time needed by Graal to process large-sized git repositories. This will allow us to better understand the scope of work that is needed and where optimization efforts should be applied in the future.

August 14 to August 19: This would work as a *buffer period* for suggestions and improvements that are needed in the implementation and in case there's a delay in any point in the described timeline.

Future Work and After GSoC ends: I would love to maintain the project once GSoC '19 ends. Maintaining Grimoirelab tools including the Graal project and enhancing other backends CoQua, CoVuln, CoDep to support a variety of languages and adding metrics related to them, some of them are:

1. **Test Coverage:** Percentage of codebase covered by developer tests employed in the software repository.

2. **Code Quality check:** Analysis related to quality checks such as Linters for other popular languages.
3. **Analysis of configuration files:** Analysing configuration files to provide insights on vulnerability and dependency of a software project.

I have no other commitments this summer except my end semester exams, which will commence and conclude during the last week of the community bonding period. With a proper schedule and communication, that won't be a problem.

About Me:

Nishchith Shetty | Github: inishchith
Email: inishchith@gmail.com
Time Zone: UTC+5:30
Phone: +91 9820501130

I'm a Junior Year student majoring in Bachelors in Computer Science from K.J Somaiya College of Engineering, Mumbai, India. I've earlier contributed to various open source projects of organizations such as CLTK, ScanCode-Toolkit, Cloud-CV and many more. My areas of interest are Backend Development, Data Analytics and Natural Language Processing.

Experience:

ScanCode-ToolKit: One of the tools which Graal CoLic backend leverages on, in order to extract license related information. Worked on limiting the number of URLs reported from licenses of software development repositories to a set threshold using the plugin architecture which was then in the development phase, added some documentation related to usage, STMicroelectronics Licence, and Code of Conduct.

Soulskill Research: Worked on Query Expansion using Word2Vec in order to improve the search interface of the product using the entire Wikipedia Data in different languages, Deployed a web scraper using Kubernetes and Docker.

Related Work:

I've completed all of 10 **Microtasks** listed on the Idea Page. I've been fairly comfortable with Graal's codebase and have added improvements & enhancements such as Flake8 analyzer to the **CoQua** backend, fixed failing tests due to missing executables and have updated the language support for **CoCom** backend with Lua & GoLang. I have also been an active member in the discussion that happens on the **Ideas page** and resolving issues. Information related to my contributions to GrimoireLab's GitHub projects can be found **here**.