Software City: Git Workflow Project

Jonas Länzlinger

July 3, 2024

Project Description

The idea is to extend the preexisting *Software City* tool, such that it becomes usable for visualizing the development history of GitHub repositories, combined with optional bio-metric data. At the moment, users need to upload a csv-file with either static software metrics or eye-tracking data. By integrating the *Software City* tool in the CI/CD pipeline of GitHub, users are not required to manually provide data files for visualizing their projects. We have specified two seperate use cases for the implementation. (1) The user integrates a GitHub Actions workflow in his repository that computes static software metrics, and registers the repository with the *Software City* tool. (2) In the second way that we plan, how users should be able to visualize GitHub repositories, they don't need to preregister a repository via GitHub Actions. They can specify a GitHub url, under which the tool will fetch the data, and subsequently compute the static software metrics.

The original *Software City* tool will remain intact and the existing functionalities unchanged. The implementation of this project will include two components:

- *Software City* tool
- GitHub Actions implementation

Software City tool

The tool needs to get extended, to provide a possibility to select a (optionally preregistered) GitHub repository from a drop-down list. Additionally this will constitute a new type of visualization, because the existing visualization types are based only on either static software metrics or bio-metric data.

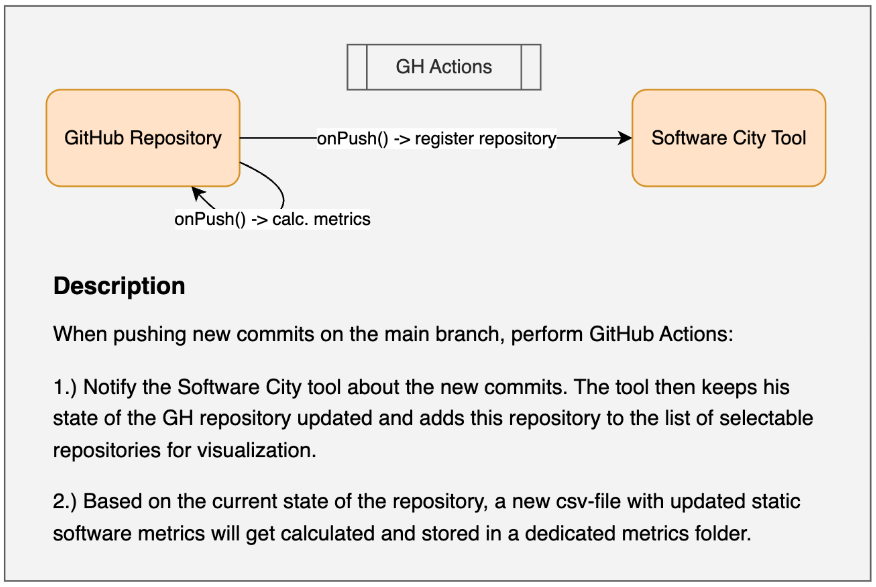
GitHub Actions implementation

The registering process of a GitHub repository will be handled by GitHub Actions. Here, a HTTP-request will be sent to the *Software City* tool. Two types of metrics need to be provided: Static software metrics, and bio-metric data. Therefore, it is necessary to implement (either in GitHub Actions directly, or via scripts) a way to create csv-files for those two metric types, including two separate folders to store them.

Workflow

The following diagrams illustrate the workflow of the finished registering and visualizing of a GitHub repository prototype (1), consisting of the three stages:

1. GitHub Actions
2. Fetching
3. Visualizing



Description

When pushing new commits on the main branch, perform GitHub Actions:

1.) Notify the Software City tool about the new commits. The tool then keeps his state of the GH repository updated and adds this repository to the list of selectable repositories for visualization.

2.) Based on the current state of the repository, a new csv-file with updated static software metrics will get calculated and stored in a dedicated metrics folder.

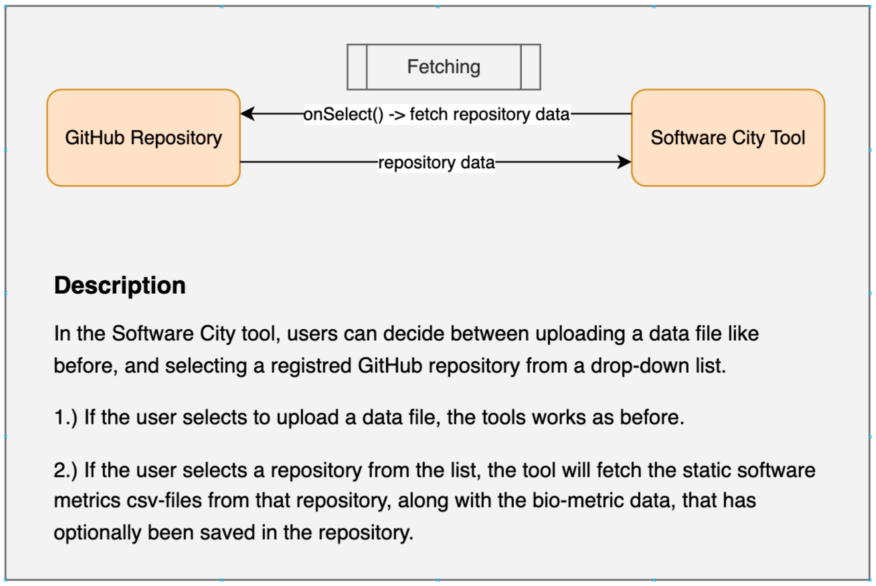
Figure 1: Workflow - GitHub Actions

Requirements - use case 1

Requirement 1: Register & fetch repository

The goal is to implement everything that is needed for the communication between a repositories' GitHub Actions and the *Software City* tool. This includes the following requirements:

- When the user pushes new commits on the main-branch, trigger GitHub Actions workflow. This workflow contains the following two functionalities.
- GitHub Actions functionality (1): Send HTTP POST-request to the *Software City* tool with a unique identifier in the request body, such that the tool can create a linkage.
- The *Software City* tool must maintain a list registered GitHub repositories.
- Once the *Software City* tool receives a HTTP POST-request from GitHub Actions, update the list of registered GitHub repositories.
- The *Software City* tool must provide an additional option in the "Upload" popup menu, which lets the user select a registered GitHub repository from a pre-populated drop-down list. Rename the "Upload" button into something more suitable.
- Once the user has selected a repository, grey out the option for selecting a upload file and the data type selection drop-down list. These fields are not needed anymore for this use case.
- When the user clicks on "Upload", fetch the static software metrics, and the bio-metric data files from the selected repository. Note: This can be achieved through 3rd party libraries, such as "GitPython" (Python) or "node-git" (JavaScript) → have a look at both options. These metrics data will get stored in the internal data store of the tool, and the rest should work as usual.
- GitHub Actions functionality (2): Create (if doesn't exist already) a folder "metrics", containing two subfolders "static", and "bio-metric". Compute a new static software metrics csv-file for each individual commit and store them in the respective folder. It is important to include a field "timestamp" with the timestamp of the respective commit → to link them later to a snapshot in the software city. For now assume that this file is provided manually. It is furthermore assumed that bio-metric data can be obtained between each commit (not each push on main). Therefore, a bio-metric data csv-file is linked to one specific commit in the Git history.



Description

In the Software City tool, users can decide between uploading a data file like before, and selecting a registred GitHub repository from a drop-down list.

1.) If the user selects to upload a data file, the tools works as before.

2.) If the user selects a repository from the list, the tool will fetch the static software metrics csv-files from that repository, along with the bio-metric data, that has optionally been saved in the repository.

Figure 2: Workflow - Fetching

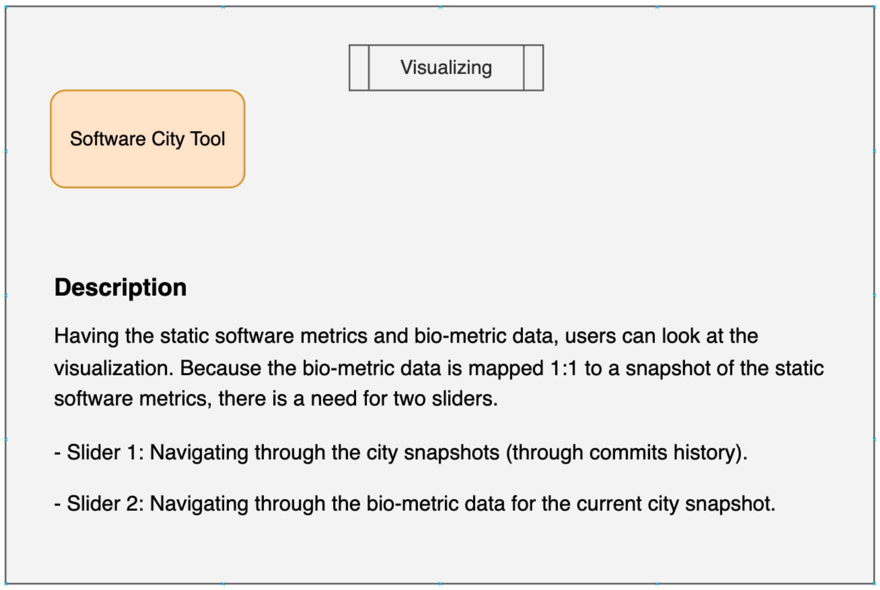
Requirement 2: Combine metric types in visualization (low priority)

For the second requirement, I want to implement the possibility to combine the visualization of static software metrics (e.g. data type ”Java Source Code”) and bio-metric data (e.g. data type ”Eye-tracking (Java Source Code)”). Because the bio-metric data is always directly linked to one specific software city snapshot, it is not possible to navigate through the static software metrics and the bio-metric data at the same time. Therefore, a second slider needs to be implemented for this visualization type, that takes the software city snapshot, and slides through the bio-metric data that have been collected for this exact snapshot. When changing the snapshot, a new set of bio-metric data will be considered for the second slider.

- If the user wants to visualize a GitHub repository, the displayed visualization type will be with combined metric types. In that case, show the visualization view with two sliders inside the top navigation bar.
- Note: To program the combined metrics visualization, I imagine I don’t have to code a lot new things. For the static software metrics, the code for the data type ”Java Source Code” that is responsible for splitting the metrics data into different snapshots (”TreeOfBuilding”) has to be adapted. Following that, it should be possible to enhance the static software metrics data with the bio-metric data, and take the implementations from the data type ”Eye-tracking (Java Source Code)” as a starting point.
- For every snapshot, a new ”ModelTree” element needs to be created.
- When exiting the visualization remember to clear the whole data store → potentially the logic there needs to be updated.
- When assigning the metaphor to attribute mapping, think about potential error scenarios when e.g. no bio-metric data are assigned to height or color metaphors.

Requirement 3: Calculate metrics

So far, the static software metrics are assumed to be provided manually by the user. The third requirement is centered around finding a suitable plugin or scripts that are invokeable from GitHub Actions, in order to generate the desired static software metric files. Requirements for the target format of the metric file are defined as follows.:



Having the static software metrics and bio-metric data, users can look at the visualization. Because the bio-metric data is mapped 1:1 to a snapshot of the static software metrics, there is a need for two sliders.

- Slider 1: Navigating through the city snapshots (through commits history).

- Slider 2: Navigating through the bio-metric data for the current city snapshot.

Figure 3: Workflow - Visualizing

- Each line in the csv-file constitutes a class.
- Each attribute needs to be non-null.
- Include a timestamp attribute.
- Include at least two additional attributes, in order to be able to display buildings in the visualization.

Note: This feature is not of the highest priority, as it is only extracting software metrics, which is already done previously by lots of other people. Nevertheless, this could cost quite some time to find a suitable tool.

Requirements - use case 2

Requirement 1: Fetch repository

The use case 2 is much simpler. We don't need to implement anything on the GitHub Actions side. The idea is to provide a url, fetch the repository data from GitHub, and compute the metrics ourselves.

- Create a new popup window, that asks the user for a GitHub repository url. When clicking on "Fetch", the tool fetches the data from GitHub.

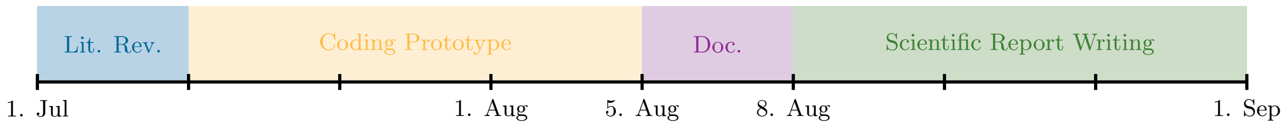
Requirement 2: Calculate metrics

As in the use case 1, we want to calculate static software metrics. But instead of doing that on GitHub via scripts, we do it ourselves with 3rd party libraries (if possible).

- Search for a 3rd party library (or potentially an IntelliJ plugin) that can compute static software metrics based on the fetched repository data from GitHub.
- Compute the metrics data and store them accordingly in the internal data store, such that the following visualization workflow doesn't need to be changed.

Roadmap

A rough roadmap for this project is outlined below. I start doing some literature research to identify what we plan to do differently from what is already been published. After the 3-4 weeks of iterative coding on the prototype, one week of each documenting is reserved, before the last 4 weeks of writing a scientific report of some sort.


Lit. Rev.
Coding Prototype
Doc.
Scientific Report Writing
1. Jul
1. Aug
5. Aug
8. Aug
1. Sep