Process Mining and Extending the Kaiaulu API

github.com/sailuh/kaiaulu

Ryan Seng, Connor Narowetz
Kaiaulu
Sponsor: Carlos Paradis
ICS 496 Capstone Project – Spring 2025

Introduction

What is Kaiaulu?

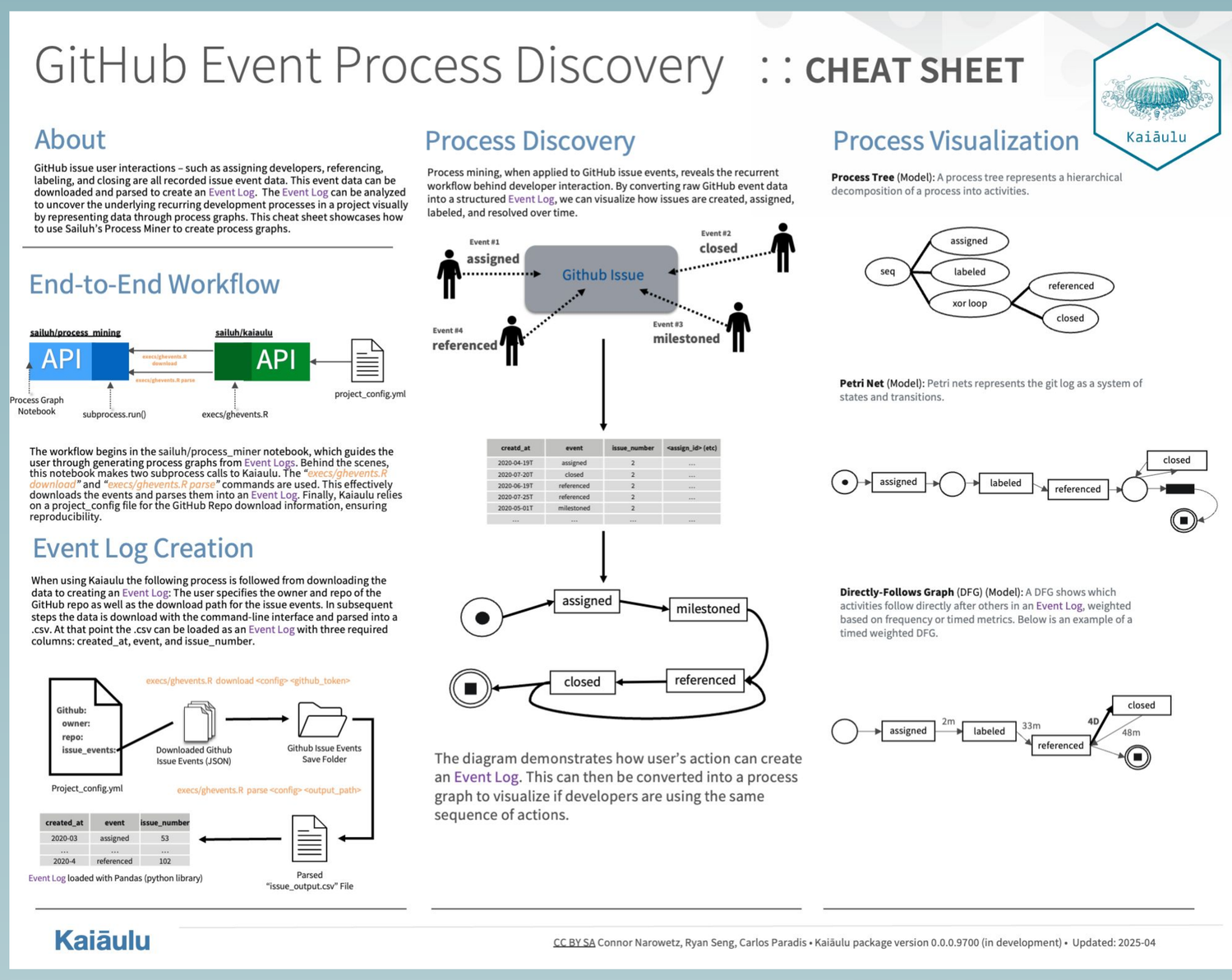
Kaiaulu is an R package and common interface that helps with understanding evolving software development communities, and the artifacts (gitlog, mailing list, files, etc.) which developers collaborate and communicate about. It allows developers to get a better understanding of the software.

Objectives

1. Create a new Python Process Mining interface to create process graphs from GitHub issue events
2. Add a new Kaiaulu capability to download Pull Request Code Review Comments and Change Requests
3. Cleanup and document Python Sentiment Analysis thesis
4. Create cheat sheets that could serve as a quick-start guide for newcomers on our developments
5. Document features and endpoints added to Kaiaulu for future development through self-generated docs

Challenges

- Unfamiliar with R and had to learn the project's structure, dependencies, and prior student contributions
- New to the GitHub REST API, which is complex and required research to find and use the correct endpoints
- No prior experience with PM4Py or process mining techniques
- Faced the challenge of learning and integrating several unfamiliar technologies at once
- Translating unstructured GitHub event data into a structured event log format required trial and error
- Ensuring data quality and consistency across multiple API pulls and file formats (JSON, CSV, etc)
- Difficulty understanding and documenting Sentiment Analysis thesis



GitHub Event Process Discovery : : CHEAT SHEET

About

GitHub issue user interactions – such as assigning developers, referencing, labeling, and closing are all recorded issue event data. This event data can be downloaded and parsed to create an Event Log. The Event Log can be analyzed to uncover the underlying recurring development processes in a project visually by representing data through process graphs. This cheat sheet showcases how to use Sailuh's Process Miner to create process graphs.

End-to-End Workflow

The workflow begins in the sailuh/process_miner notebook, which guides the user through generating process graphs from Event Logs. Behind the scenes, this notebook makes two subprocess calls to Kaiaulu. The "*execs/ghevents.R download*" and "*execs/ghevents.R parse*" commands are used. This effectively downloads the events and parses them into an Event Log. Finally, Kaiaulu relies on a project_config file for the GitHub Repo download information, ensuring reproducibility.

Event Log Creation

When using Kaiaulu the following process is followed from downloading the data to creating an Event Log: The user specifies the owner and repo of the GitHub repo as well as the download path for the issue events. In subsequent steps the data is download with the command-line interface and parsed into a .csv. At that point the .csv can be loaded as an Event Log with three required columns: created_at, event, and issue_number.

created_at	event	issue_number
2020-03	assigned	53
...	...	...
2020-4	referenced	102

Event Log loaded with Pandas (python library)

Process Discovery

Process mining, when applied to GitHub issue events, reveals the recurrent workflow behind developer interaction. By converting raw GitHub event data into a structured Event Log, we can visualize how issues are created, assigned, labeled, and resolved over time.

creatd_at	event	issue_number	<assign_id> (etc)
2020-04-19T	assigned	2	...
2020-07-20T	closed	2	...
2020-06-19T	referenced	2	...
2020-07-25T	referenced	2	...
2020-05-01T	milestoned	2	
...	...	...	...

The diagram demonstrates how user's action can create an Event Log. This can then be converted into a process graph to visualize if developers are using the same sequence of actions.

Process Visualization

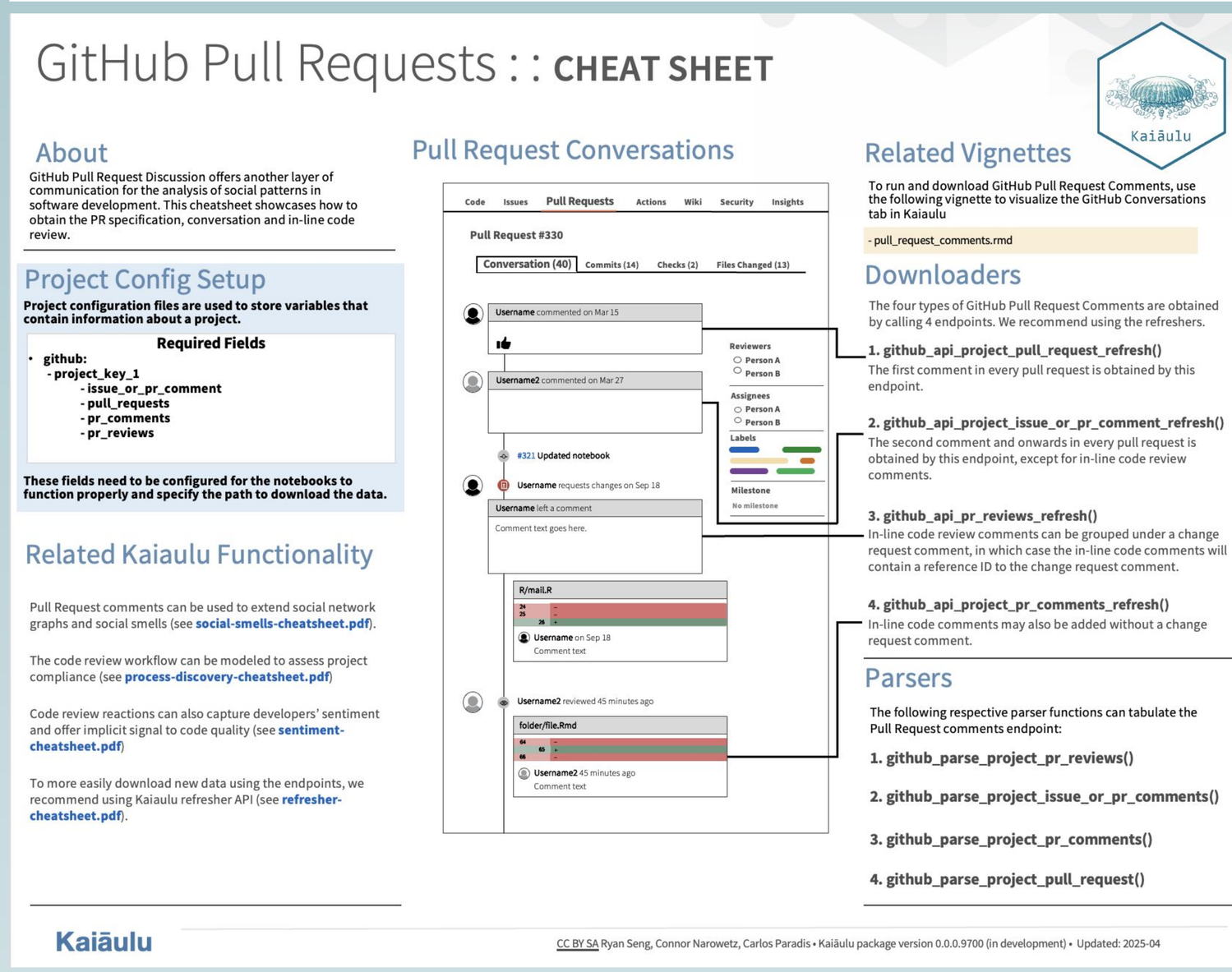
Process Tree (Model): A process tree represents a hierarchical decomposition of a process into activities.

Petri Net (Model): Petri nets represents the git log as a system of states and transitions.

Directly-Follows Graph (DFG) (Model): A DFG shows which activities follow directly after others in an Event Log, weighted based on frequency or timed metrics. Below is an example of a timed weighted DFG.

Kaiāulu

CC BY SA Connor Narowetz, Ryan Seng, Carlos Paradis • Kaiāulu package version 0.0.0.9700 (in development) • Updated: 2025-04



GitHub Pull Requests : : CHEAT SHEET

About

GitHub Pull Request Discussion offers another layer of communication for the analysis of social patterns in software development. This cheatsheet showcases how to obtain the PR specification, conversation and in-line code review.

Project Config Setup

Project configuration files are used to store variables that contain information about a project.

Required Fields

- **github:**
  - **project_key_1**
    - **issue_or_pr_comment**
    - **pull_requests**
    - **pr_comments**
    - **pr_reviews**

These fields need to be configured for the notebooks to function properly and specify the path to download the data.

Related Kaiaulu Functionality

Pull Request comments can be used to extend social network graphs and social smells (see **social-smells-cheatsheet.pdf**).

The code review workflow can be modeled to assess project compliance (see **process-discovery-cheatsheet.pdf**)

Code review reactions can also capture developers' sentiment and offer implicit signal to code quality (see **sentiment-cheatsheet.pdf**)

To more easily download new data using the endpoints, we recommend using Kaiaulu refresher API (see **refresher-cheatsheet.pdf**).

Pull Request Conversations

Related Vignettes

To run and download GitHub Pull Request Comments, use the following vignette to visualize the GitHub Conversations tab in Kaiaulu

- pull_request_comments.rmd

Downloaders

The four types of GitHub Pull Request Comments are obtained by calling 4 endpoints. We recommend using the refreshers.

1. github_api_project_pull_request_refresh()
The first comment in every pull request is obtained by this endpoint.

2. github_api_project_issue_or_pr_comment_refresh()
The second comment and onwards in every pull request is obtained by this endpoint, except for in-line code review comments.

3. github_api_pr_reviews_refresh()
In-line code review comments can be grouped under a change request comment, in which case the in-line code comments will contain a reference ID to the change request comment.

4. github_api_project_pr_comments_refresh()
In-line code comments may also be added without a change request comment.

Parsers

The following respective parser functions can tabulate the Pull Request comments endpoint:

1. github_parse_project_pr_reviews()

2. github_parse_project_issue_or_pr_comments()

3. github_parse_project_pr_comments()

4. github_parse_project_pull_request()

Kaiāulu

CC BY SA Ryan Seng, Connor Narowetz, Carlos Paradis • Kaiāulu package version 0.0.0.9700 (in development) • Updated: 2025-04

Solution

Technologies

Python – for scripting, data transformation, and process mining.

➔ **Pm4py** – provided package for process mining models and algorithms.

R & R Studio – as the development environment for working on Kaiaulu.

GitHub REST API – to fetch GitHub issue events and metadata

Git & GitHub platform – used for version control, project management, issue tracking

Tasks Accomplished

1. Implemented four GitHub comment endpoints with a refresher function to refresh data requests.
2. Created GitHub Issue event data download and parser executable CLI to be called by process mining interface on runtime.
3. Created interface to create multiple different process graphs including Petri Nets, Process Trees, BPMN (Business Process Model and Notion, and Directly-Follows Graphs).
4. Documented features and created cheatsheets for future development.
5. Documented and refactored Sentiment Analysis capability thesis.

Learnings & Conclusion

Throughout this project, wel learned to navigate Python and R in the context of a large project, using a new IDE (R Studio). We gained hands-on experience with the PM4Py library and process mining techniques being exposed to new algorithms and models we have never seen before. We also became proficient with the GitHub REST API, overcoming its complexity and working with various endpoints. Additionally, we learned how to work with a large existing code base and architectural practices for organizing code and collaborating effectively. This project significantly deepened our understanding of version control in a collaborative environment, particularly through structured workflows in Git and GitHub. These practices not only improved our technical fluency but also positioned us well for future work involving large-scale data analysis and process mining efforts. Together we believe we contributed successfully to the Kaiaulu R package and acquired critical skills necessary for our futures working with software.