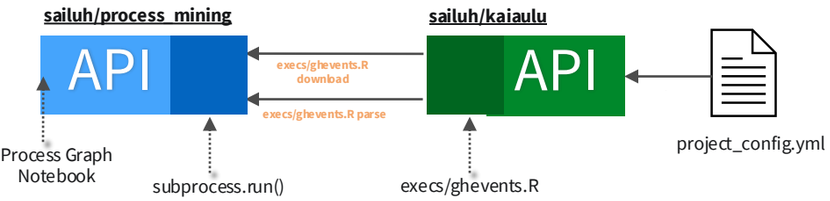
GitHub Event Process Discovery : : CHEAT SHEET

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.

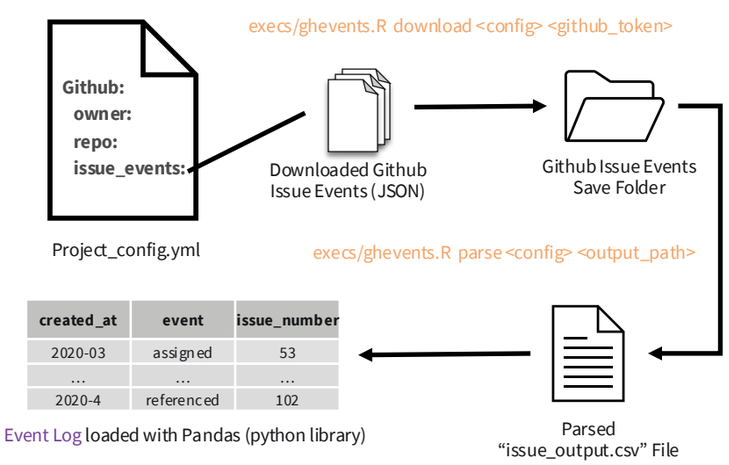
End-to-End Workflow



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.

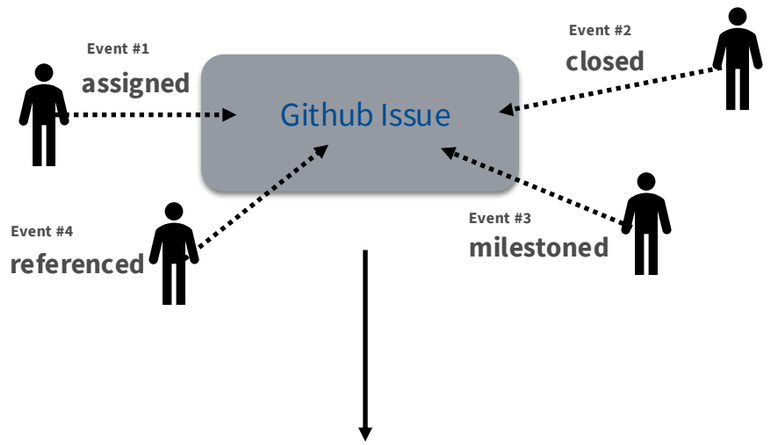
Event Log Creation

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.

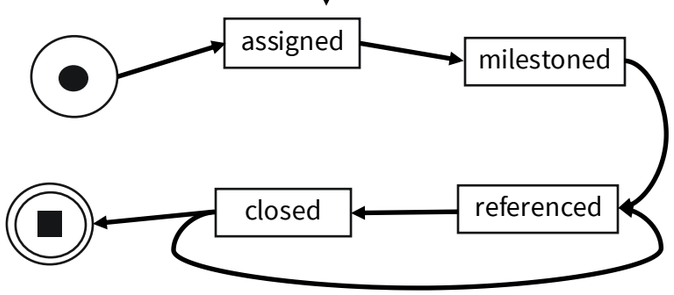


Process Discovery

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.



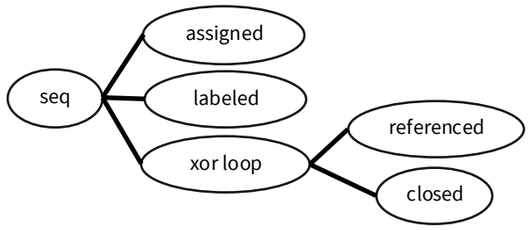
creatd_at	event	issue_number	<assign_id> (etc)
2020-04-19T	assigned	2	…
2020-07-20T	closed	2	…
2020-06-19T	referenced	2	…
2020-07-25T	referenced	2	…
2020-05-01T	milestoned	2	
…	…	…	…



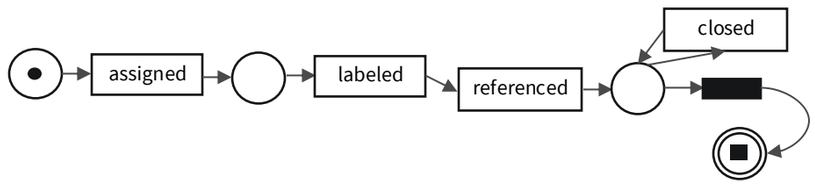
The diagram demonstrates how user's action can create an Event Log. This can then be converted into a process graph to visualize if developers are using the same sequence of actions.

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.