

Software Artifact Analyzer

User Guide

Version 1.0.0 July 2024

1 Introduction	2
2 Getting Started with Software Artifact Analyzer (SAA)	2
2 Menubar	5
2.1 Project Menu	5
2.1.1 New Project Modal	5
2.1.1 About Project	7
2.2 File Menu	7
2.3 Edit Menu	8
2.3 View Menu	10
2.4 Highlight Menu	11
2.5 Layout Menu	13
2.6 Help Menu	15
2.7 Data Menu	15
2.8 Context Menus	16
3 Toolbar	22
4 Objects	23
4.1 Object Inspection Tab	23
4.2 Object Statistics Tab	24
4.3 Object Report Tab	24
4.3.1 Report Issue	24
4.3.2 Report Pull Request	25
4.3.3 Report Commit and File	26
4.3.5 Report Developer	27
4.4 Object Queries Tab	27
4.4.1 Expert Recommendation For File Node	27
4.4.2 Reviewer Recommendation For Pull Request Node	28
5 Map	29
5.1 Filter by Node/Edge Type	29
5.2 Query by Rule	30
5.3 Group Nodes	31
6 Database	32
6.1 General Queries	32
6.2 Custom Queries	32
6.2.1 Get Commits of Developer	32
6.2.2 Get Issue Anomalies	32
6.2.4 Get Issue Anomaly Statistics	34

1 Introduction

This guide will show you how to use Software Artifact Analyzer tool (SAA), which is designed to enhance efficiency in software development processes. SAA meticulously analyzes various software artifacts, including source code files, pull requests, issues, and commits, and their interconnected relationships with developers within a project. It uses an advanced drawing canvas with features such as context menus and visual cues to facilitate this comprehensive analysis. SAA uses a structured approach, and at its core is a toolkit named Visual [1], which provides a solid foundation for its functionalities, such as the user-friendly interface for developers to interact with the system. With this interface, developers can explore and make informed decisions by leveraging insights derived from the comprehensive artifact traceability graph.

The SAA is designed to cater to projects utilizing GitHub as a git source and Jira as a issue tracking tool. Also, if you want to customize SAA and use it with other CASE tools, following a structured approach will make it easier to integrate those additional tools. The tool engine integrates seamlessly with GitHub and Jira APIs, allowing for data access and interaction with these platforms. SAA is also equipped with core functions like finding the proper reviewer for the pull request or experts for a file, as well as detecting the bug process smells in the project. In essence, SAA should be a valuable asset for software development teams as it provides a clear and intuitive pathway to delve into software artifacts, understand their relationships, and make informed decisions throughout the development lifecycle.

As many features are inherited from the base toolkit Visual [1], you might want to refer to [Visual's User's Guide](#) for details of most generic features.

2 Getting Started with Software Artifact Analyzer (SAA)



Figure 1: CASE tools integration flow

To start using SAA, you first need to install the Software Artifact Analyzer GitHub Application. Visit the [GitHub application page](#) and install the [SAA application](#) on your GitHub user account. Select a specific project or grant permission to all projects.

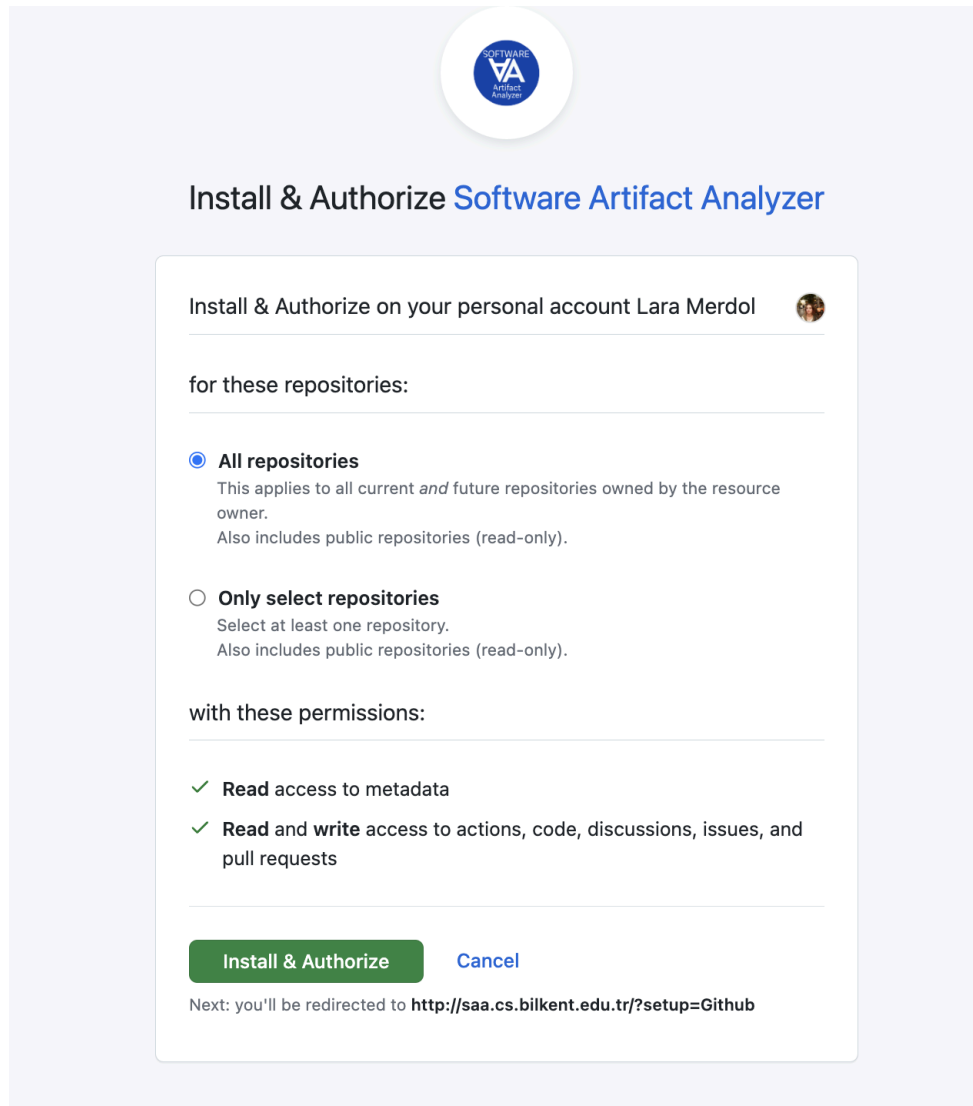


Figure 2: SAA GitHub application repository selection page

After installation, you will be redirected to SAA for further authentication from Jira. Click on "Jira" to proceed to the Jira authentication page and grant permission to the Jira instance you want to use with SAA.

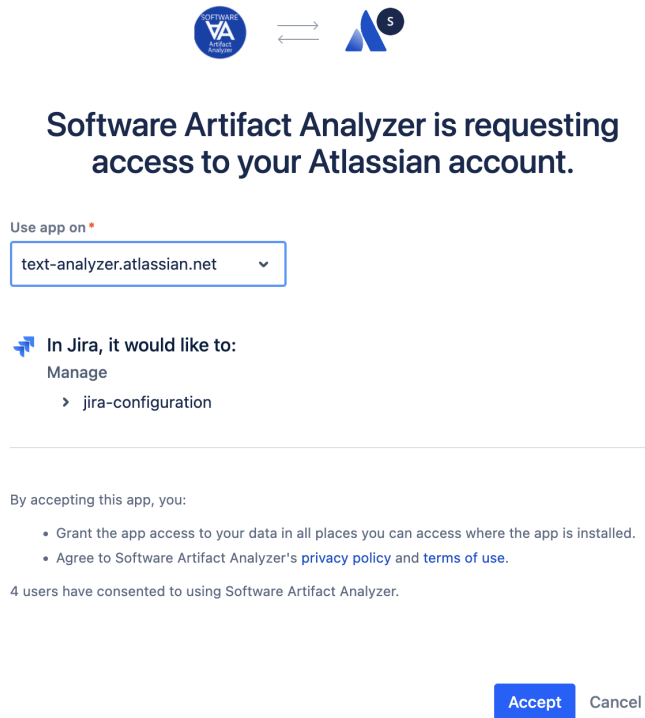


Figure 3: SAA Atlassian application Jira instance selection page

Next, you will be redirected to the SAA Neo4j instance credentials page. Here, you can either continue with the default option or provide credentials for your own Neo4j instance.

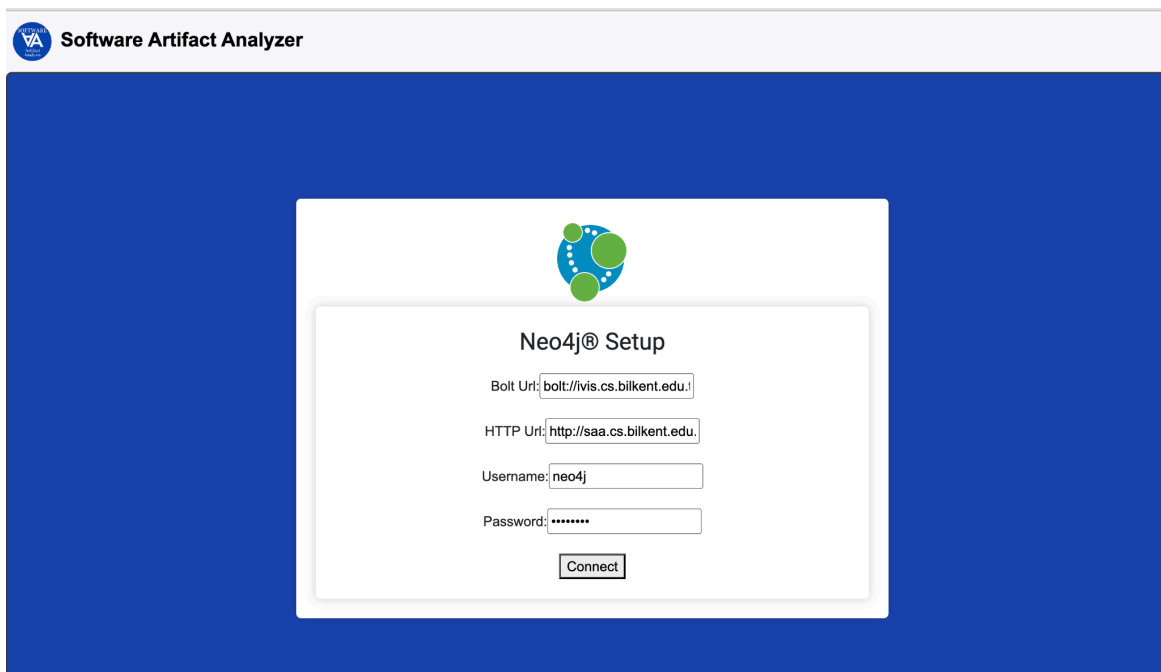


Figure 4: SAA Neo4j instance entering page

After selecting your Neo4j instance, you will be redirected back to the SAA page.

2 Menubar

Menubar contains options for performing project, file, editing, view, highlight, and layout operations. A frequently used subset of these operations also appears in the Toolbar for convenience.

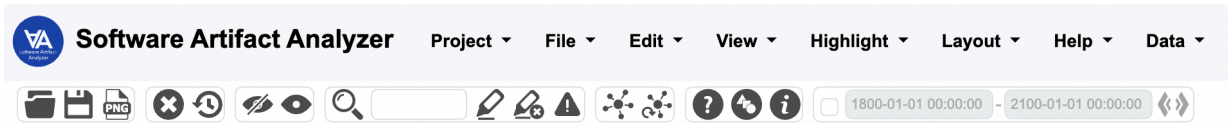


Figure 5: Menubar

2.1 Project Menu

The project menu allows users to open new projects and learn certain statistics about the current project.

2.1.1 New Project Modal

From the "Project | New..." tab one can retrieve their project data and build a new Neo4j database in 4 steps.

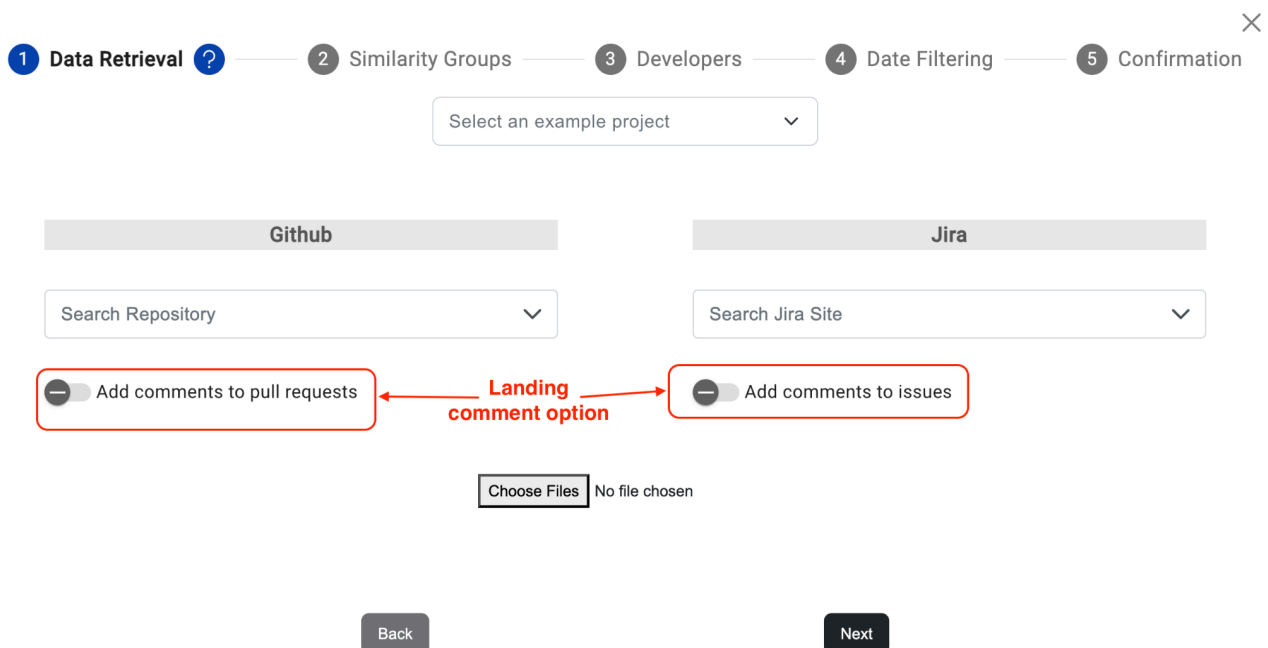


Figure 6: "Data Retrieval" step user interface

1. To retrieve project credentials for Github and Jira accounts, one should proceed as follows:
 - Navigate to the "Data Retrieval" step
 - Enter the project credentials for Github and Jira

Note: New users can test the application using open-source Apache example projects (without authentication) available under a dropdown menu.

2. To identify potential mergers among similar developers' groups (instances with the same developer), one can navigate to the "Similarity Groups" step. When reviewing the detected similar developer groups, if the user agrees with the detection, they accept the mergers by selecting the corresponding option.

Figure 7: "Similarity Groups" step user interface

3. If similar developers' groups are not identified during the "Similarity Groups" step, you can still add new similarity groups or modify existing ones from the "Developers" step.

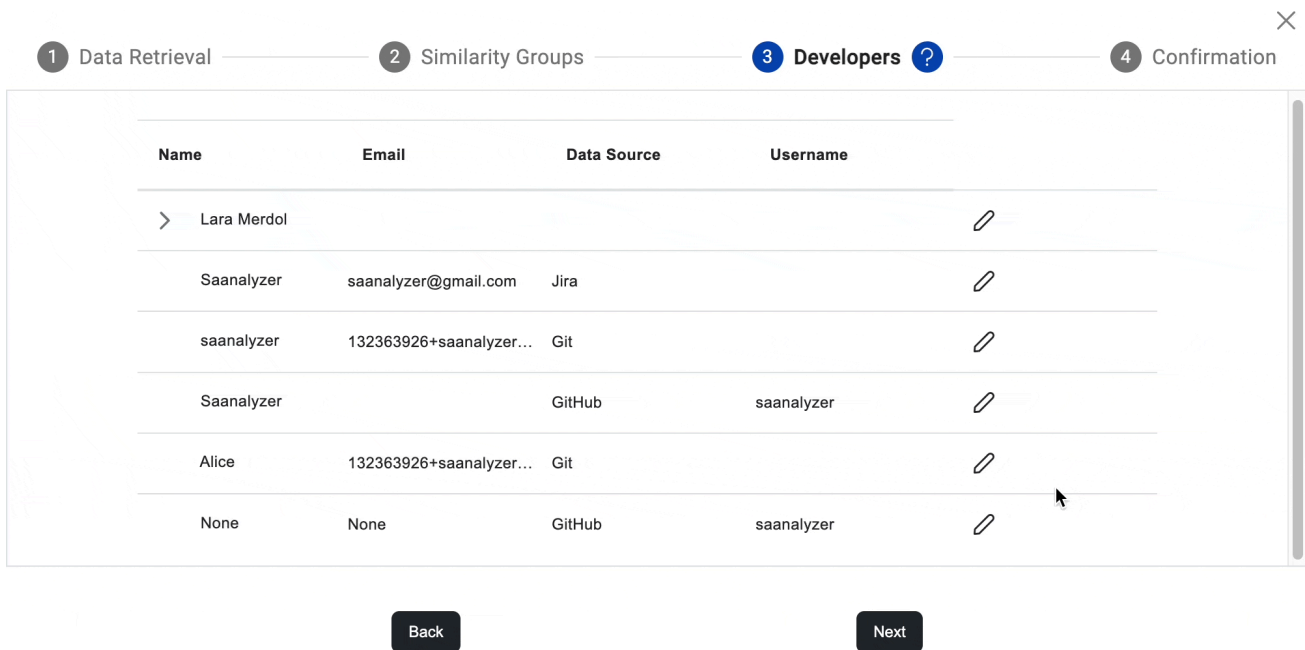


Figure 8: "All Developers" step user interface

4. SAA allows users to filter data based on a specified time span. Users can select the start and end dates, as well as the inclusion type for software artifacts. This feature helps reduce data complexity and focus analysis on specific cases, enhancing the overall user experience.

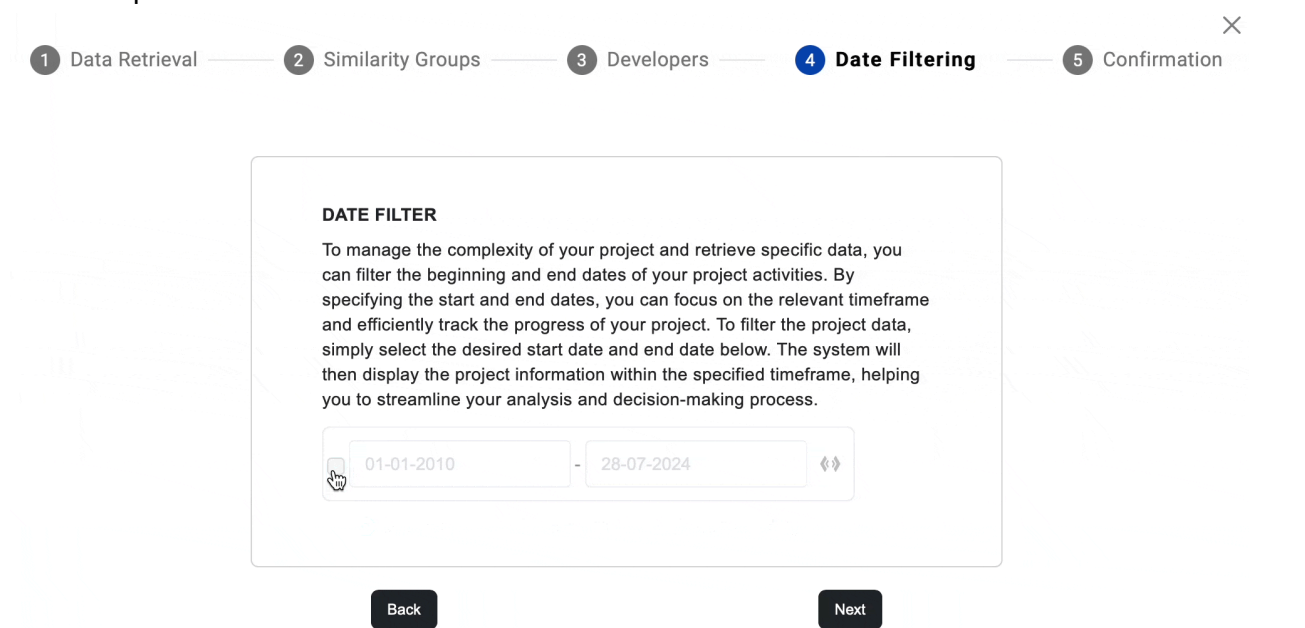


Figure 9: "Date Filtering" step user interface

5. After completing the data retrieval and cleaning process, you can confirm the data and then instruct the SAA to build the project's artifact traceability graph from the retrieved data.

2.1.1 About Project

From the "Project | About..." tab, you can learn the key details about the current project on the system. Furthermore, one can also learn about the key statistics of the project like the number of nodes and edges.

About Project ✕

Name: LaraMerdol/text_analyzer

GitHub: [url](#)

Jira: [url](#)

Statistics

Nodes: 166	Edges: 240
Developers: 3	Issues: 9
Pull Requests: 4	Commits: 92
Files: 58	

Figure 10: "About" modal

2.2 File Menu

The File menu allows one to save the current map to disk (in JSON format) and load it back when needed. Instead of the entire current map, you can also save a selected subset of elements of the map. Persistency maintains style information as well as the topology.

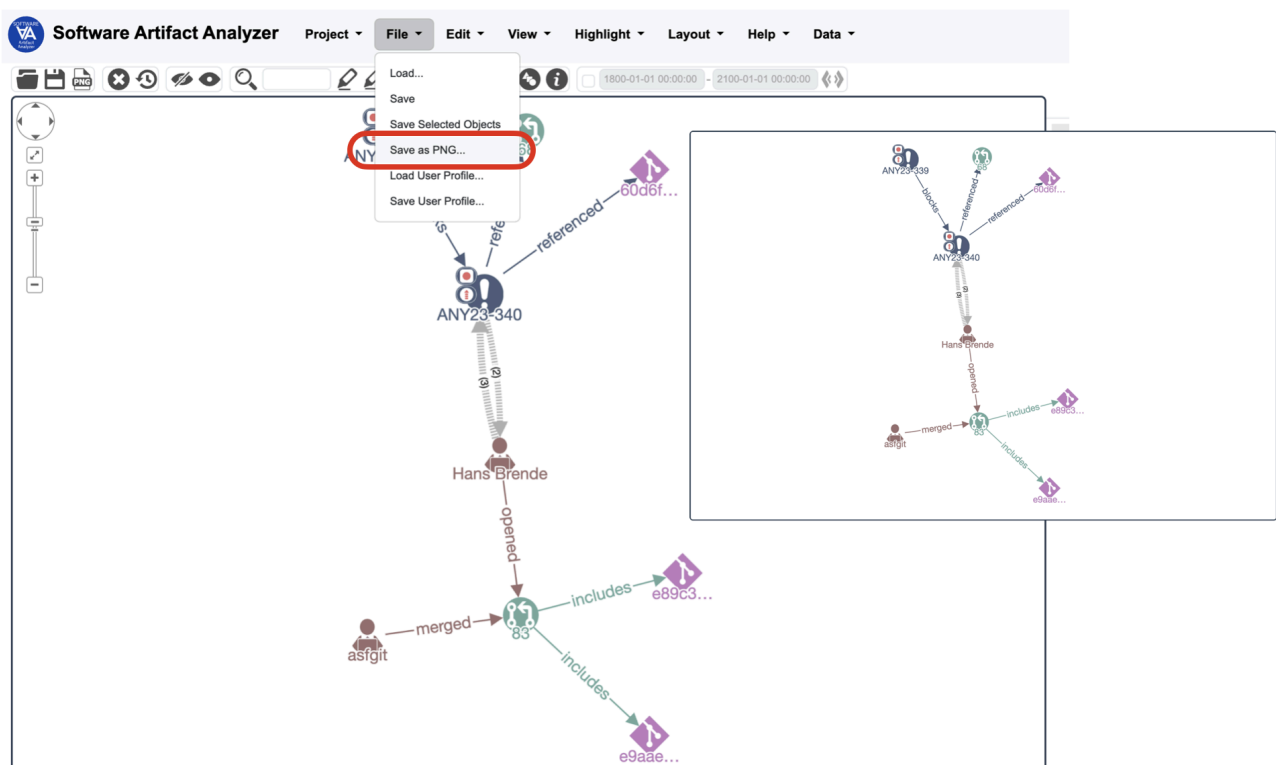


Figure11: Save as PNG

This menu also allows you to save the current map as a static PNG image.

Finally using this menu the user may save their profile (Settings, Filtering Rules, and Timebar Statistics) on disk in a proprietary format (".vall" file) to exchange with others (see "File | Load User Profile...").

2.3 Edit Menu

The Edit menu allows the user to delete selected objects from the map. Furthermore, the user can add or remove groups through this menu. Groups may be represented with either compound nodes or with clusters represented with circles upon automatic layout (see the [section on groups](#) for details of this option). See below for an example using both representations for the same map: (left) compounds, and (right) circles.

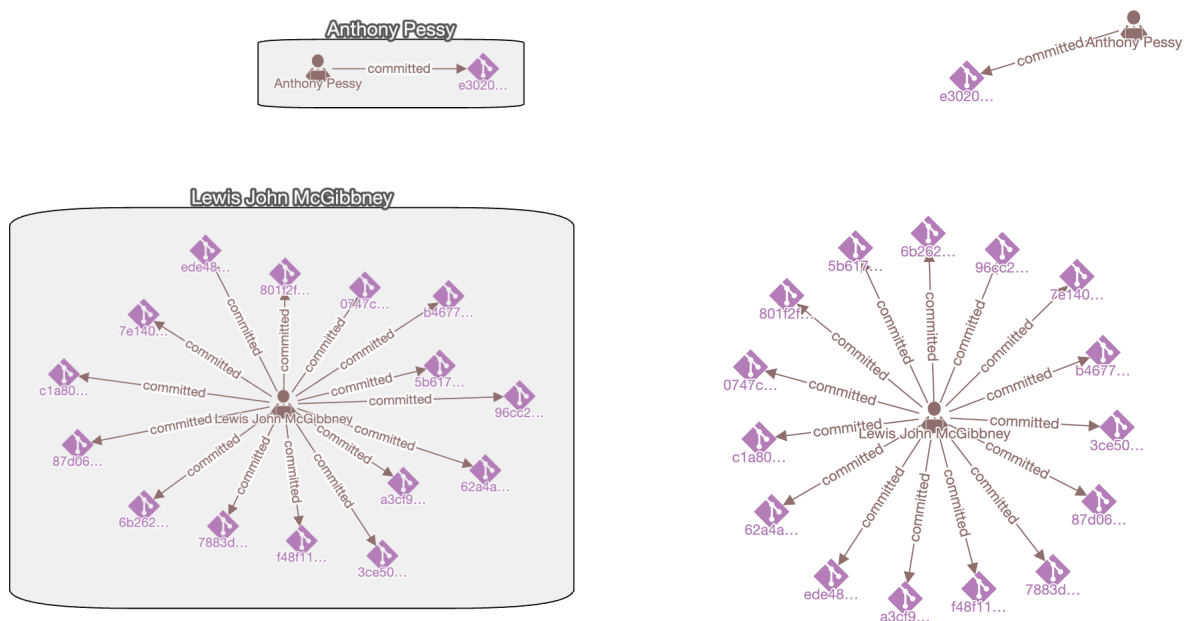



Figure 8: Group nodes representation (left) with compounds, and (right) with circles

Finally, this menu can be used to show a history of queries ("Edit | Query History") executed. When you mouse over the  icon in the dialog that pops up, you will be shown a picture of the map in your drawing canvas at that moment. If you click on a particular one, the map is reverted to the graph at that moment. This is intended to undo the latest operation(s) and go back to a previous state.

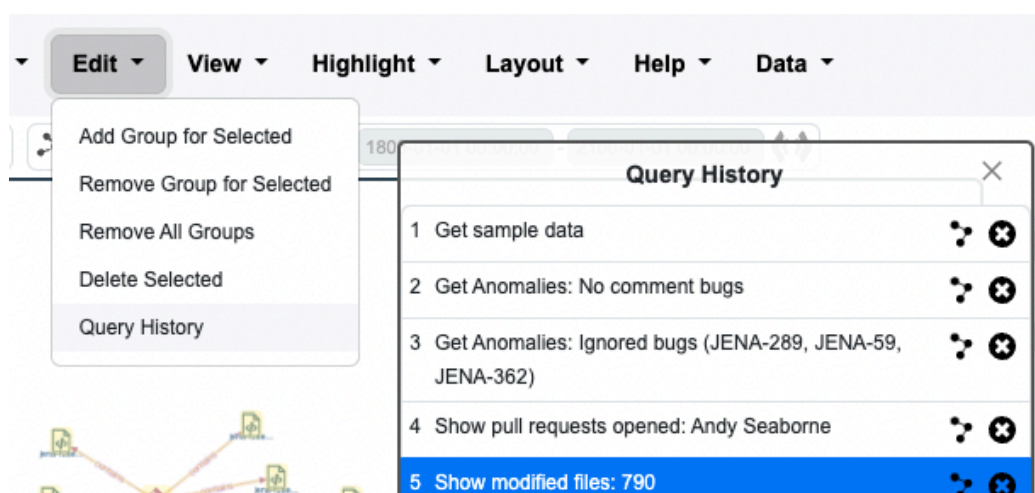


Figure 9: Query History

2.3 View Menu

The View menu contains operations to manage the complexity of the map by temporarily *hiding* selected map objects and *showing* them back when needed. Notice that this is different from deletion, which permanently deletes the map objects from the Cytoscape.js model (unless you perform a subsequent database query and bring back the same object(s)) and hence the client. Below you will find: an original map with certain objects selected after selected objects are hidden using “View | Hide Selected”, and when all map content in the client is shown with “View | Show All”.

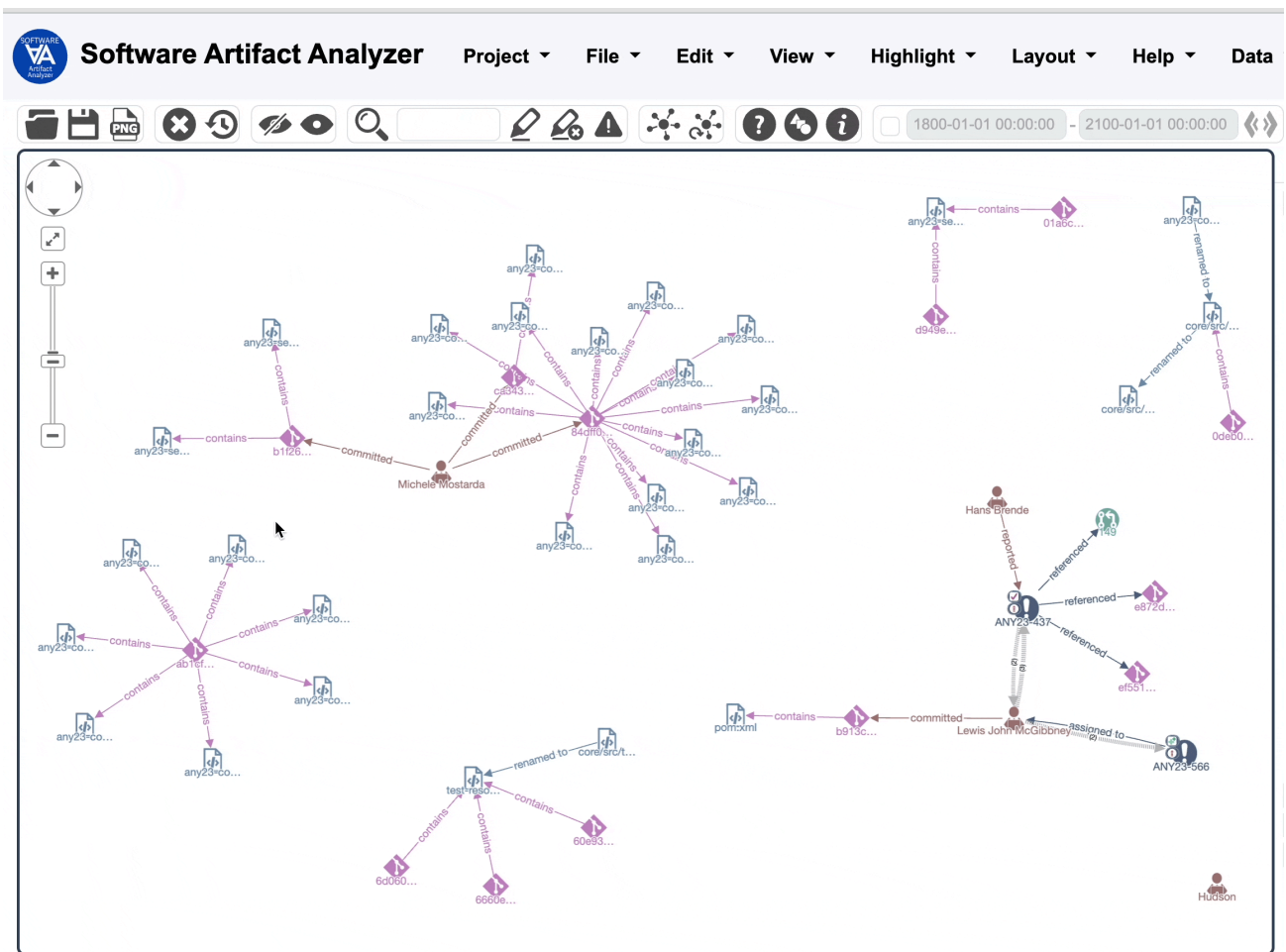


Figure 10: Hide Selected

The user can also *expand* or *collapse* (compound) nodes and (multiple) edges from the View menu. Collapsed multiple edges will be counted as “meta edges” under the Statistics tab.

Below is an example showing expanded (left) and collapsed (right) edges:



Figure 11: Expand or collapse (multiple) edges

Below is an example showing expanded (left) and collapsed (right) nodes. In order for visual cues that are used to collapse (“-” icon) or expand (“+” icon) nodes to appear, the associated compound node must be selected first.

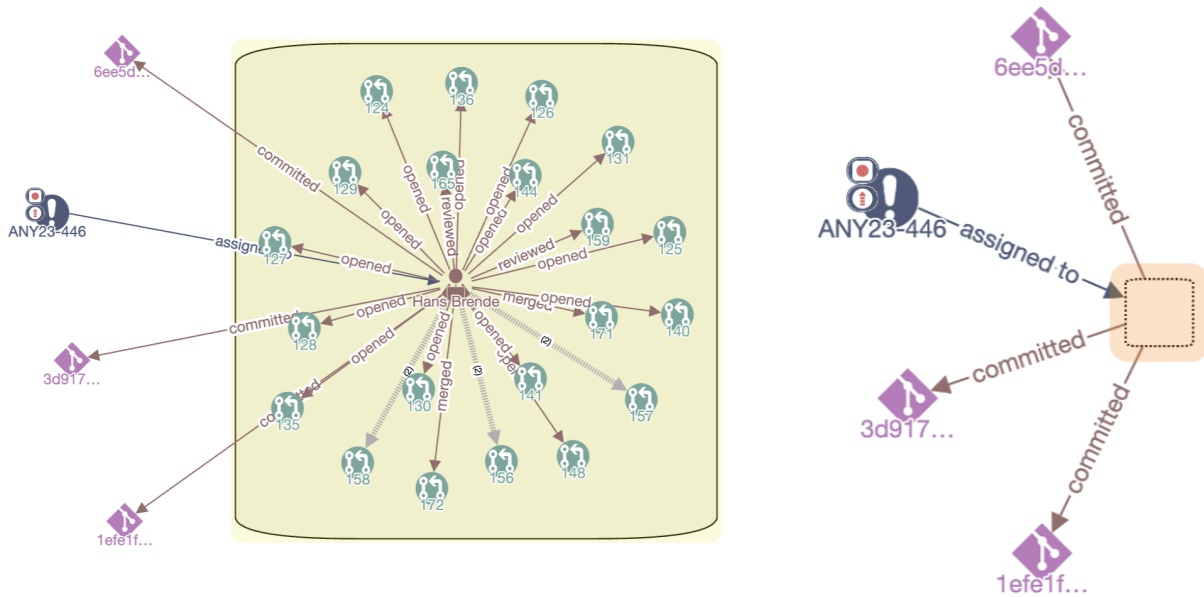


Figure 12: Expand or collapse (compound) nodes

2.4 Highlight Menu

The Highlight menu aims to draw attention to certain map objects (e.g., a group of nodes and/or edges or paths in the underlying network). The user may use this menu to highlight map objects by either selecting them or by typing a string to match in the search box located in the Toolbar. This string is searched for in labels or other properties (numbers are converted to strings before such a search is applied) of graph objects. Below is the result of searching the string "library" in a sample map. Note that the search will be case-sensitive or insensitive depending on the associated setting.



Figure 13: Result of searching the string "library"

By default, some highlight styles have been defined under Settings. The user may modify these styles or add new ones by specifying a certain color and padding. All highlights made so far could be removed from the map using "Highlight | Remove Highlights".

Check Anomalies

Users can add anomaly badges within the highlight menu to further investigate anomalies on the map by selecting "Highlight | Check Anomalies". Anomaly badges also include a tooltip. By hovering the node badges, one can also show the list of the anomalies in the issue. For a deeper understanding of the anomalies detected, refer to Section [6.2.2](#) for detailed information on their types.

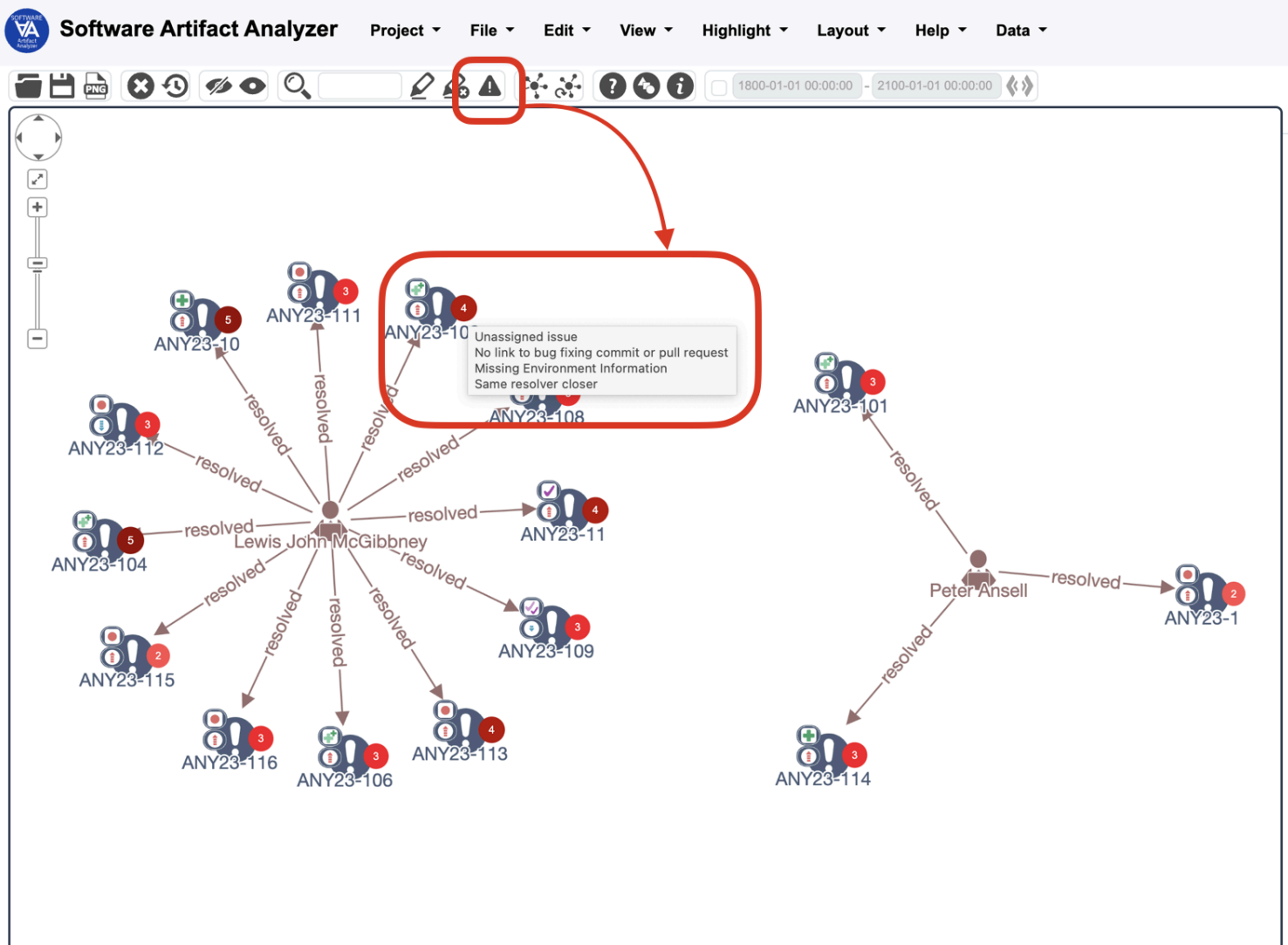


Figure 14: Adding anomaly badges

2.5 Layout Menu

The layout operations may be used to "tidy up" the layout of the current map using "Layout | Perform Layout". This operation takes the current locations of map objects into account and performs an *incremental layout* by optimizing their geometric distances to be in line with their graph-theoretic distances. However, if you think that you're not happy with the current layout and would like a new one to be calculated *from scratch*, then you should perform "Layout | Recalculate Layout". Notice that an incremental layout is applied to some interactive operations where the result of a query is *merged* into the current map instead of replacing it. When, however, the result of an operation is to *replace* the current map content, the layout is recalculated from scratch. Below is the same sample software artifact network laid out randomly, and automatically by SAA.

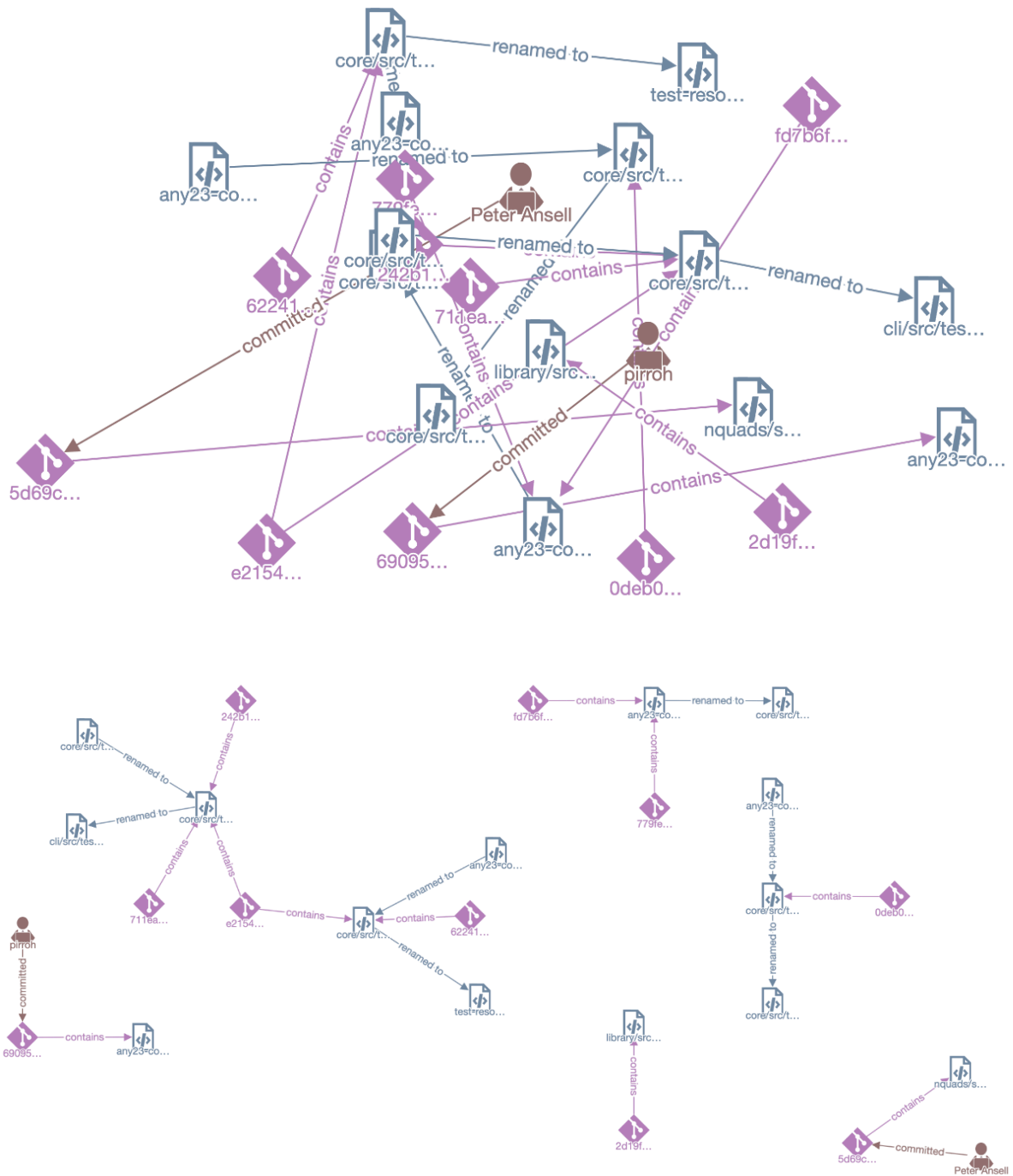


Figure 15: Sample software artifact network laid out randomly (top), and automatically (bottom)

The particular layout style used depends on whether or not the grouping of nodes is defined and how the user prefers to represent such groupings for their map. Visually uses [CiSE layout](#) [2] when circle representation is preferred, and [fCoSE layout](#) [3] otherwise. Both layout styles support performing the layout incrementally respecting current node positions and trying to preserve the user's mental map as the map topology changes.

2.6 Help Menu

Find quick assistance under the Help menu, where you can easily discover various gestures for manipulating drawings on the canvas. Additionally, access the About dialog to view the last build date. Users can also explore the Legend dialog, which provides information on the existing node and edge types within the current project.

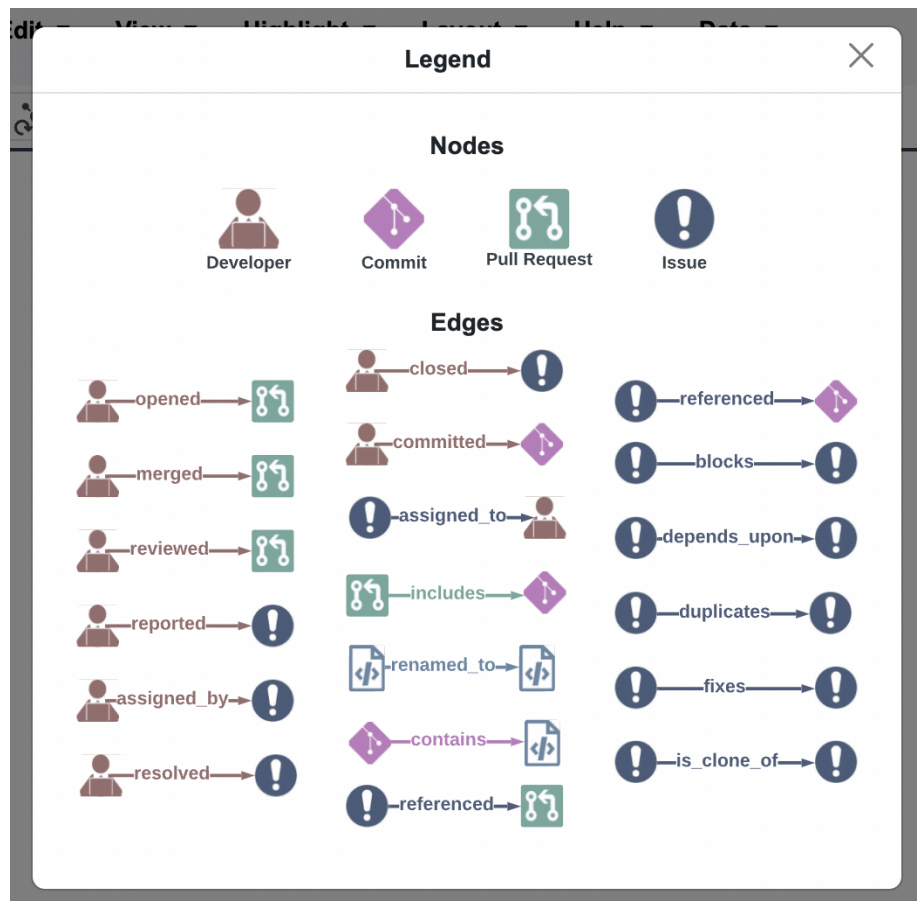


Figure 15: Legend

2.7 Data Menu

The Data menu provides easy access to load some sample data or to delete all the data on the map. This sample respects the current [time range](#). The sample data will only provide a random sample of data, which is limited to 20 nodes and edges. As a result, it will only display a portion of the graph and not all nodes and relationships.

Another method to initiate data loading is by launching SAA with query parameters. If you wish to view a specific node along with its immediate neighboring nodes, you can opt for the query parameter option by specifying the name of the artifact as a parameter, as demonstrated below.

Alternatively, you can also specify the limit number of neighbors using a query parameter. The query will retrieve the most recent neighbors up to the specified number. If you set `limit=true`, it will use the default limit of 7.

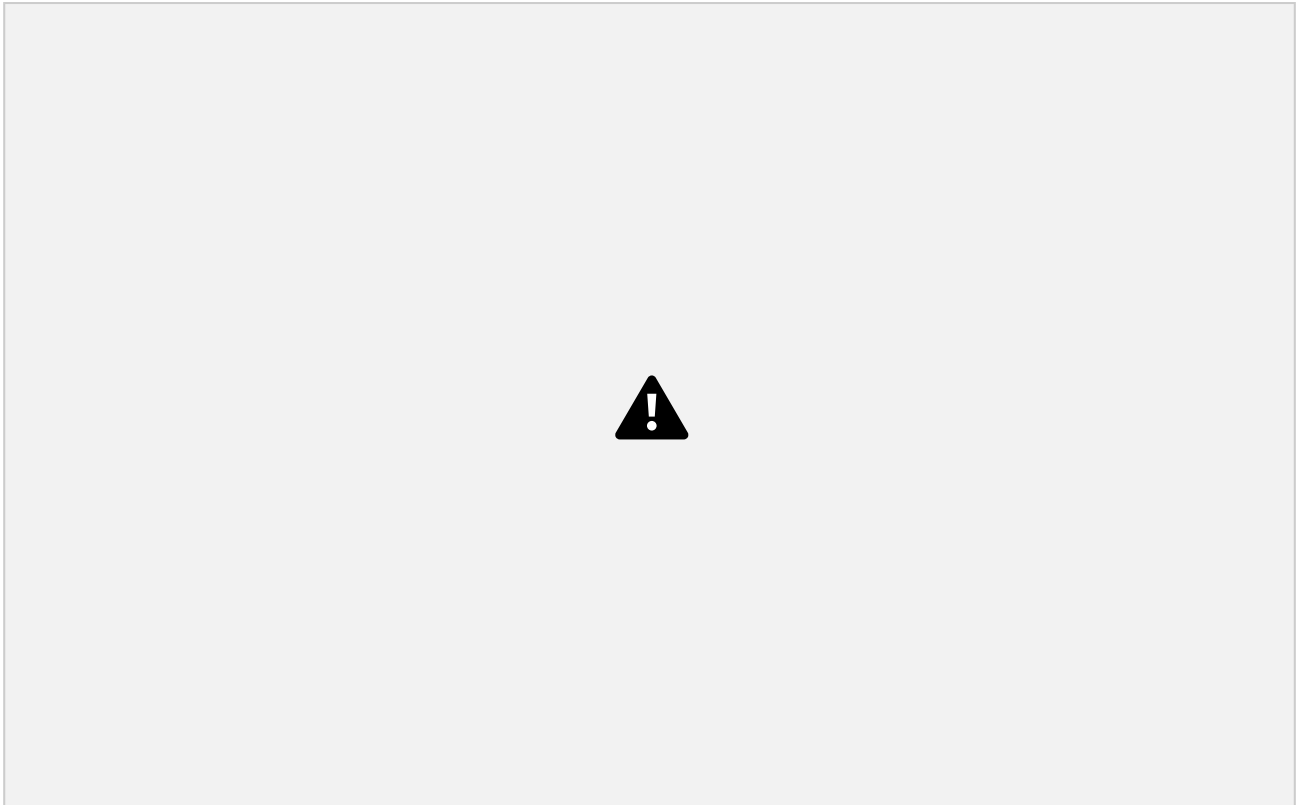


Figure 16: Data loading with query parameters

2.8 Context Menus

The other useful component of SAA is context menus that are custom to each object type. We have implemented many queries that enable the user to trace the paths between different artifacts for analyzing software processes in the project. Some of the examples can be found below. Furthermore, we add a hide option for all show options to give users a more convenient experience.

Commit's Context Menu

The user can learn a lot of information about a specific commit from the context menu of the commit type nodes. Such as who is the author, which pull request includes the commit, which files change with the commit, and who is the reviewer of the pull request that includes the commit (reviewer of the commit).

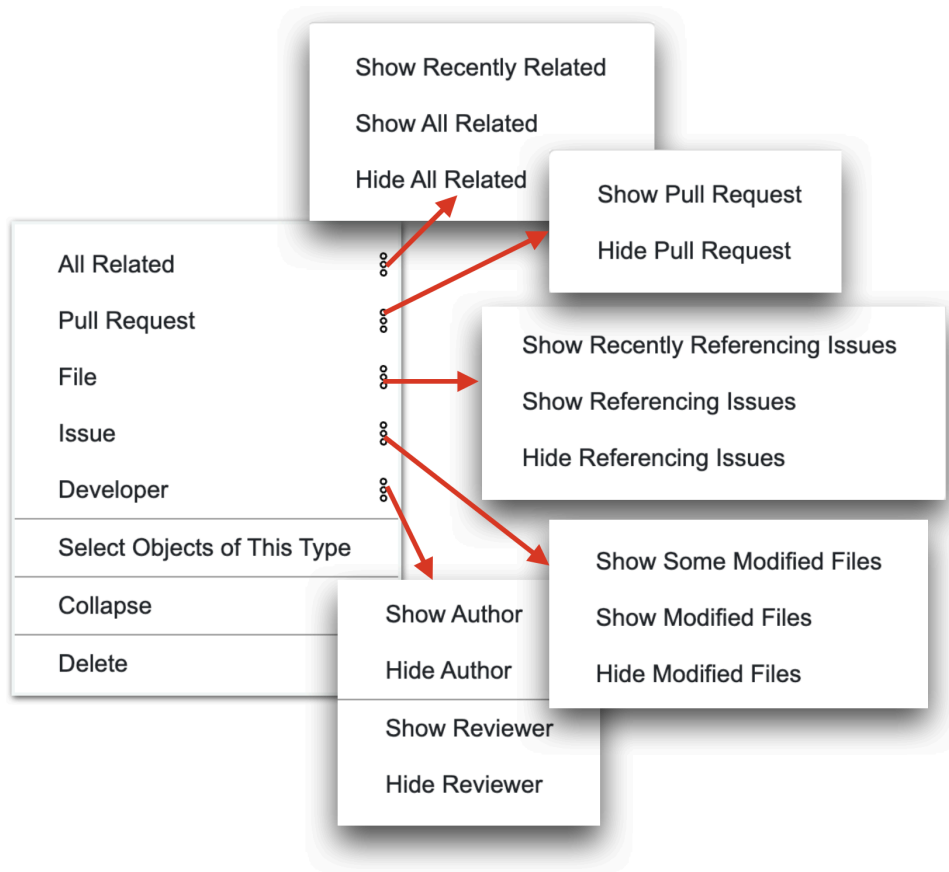


Figure 17: Commit's Context Menu

Show Commit's Reviewer: This option gets the developers who review the pull request that the selected commit included.

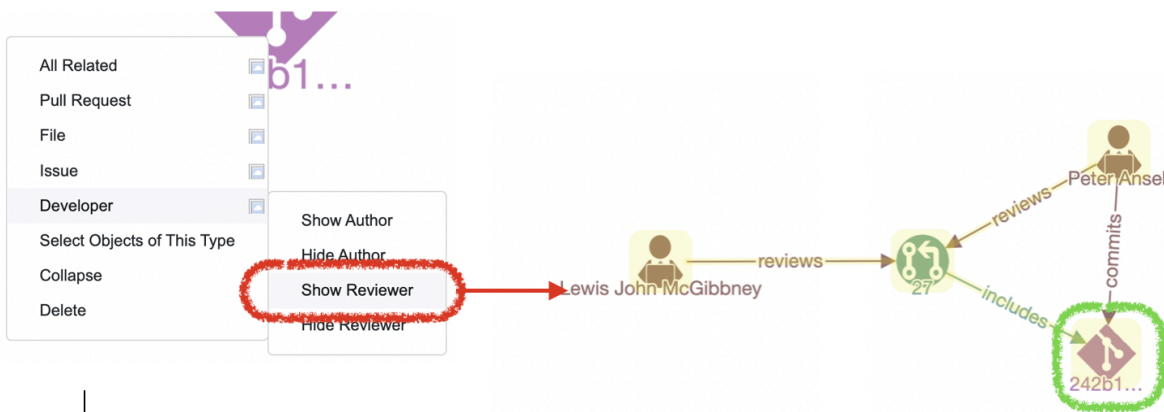


Figure 18: Commit's Context Menu | Show Reviewer

Developer's Context Menu

From the context menu of the developer-type nodes, the user can learn developer pull requests, issues, committed pull requests, committed files, and many other things.



Figure 19: Developer's Context Menu

Show Files That Developer Commits Into: This option gets files that the developer commits into.

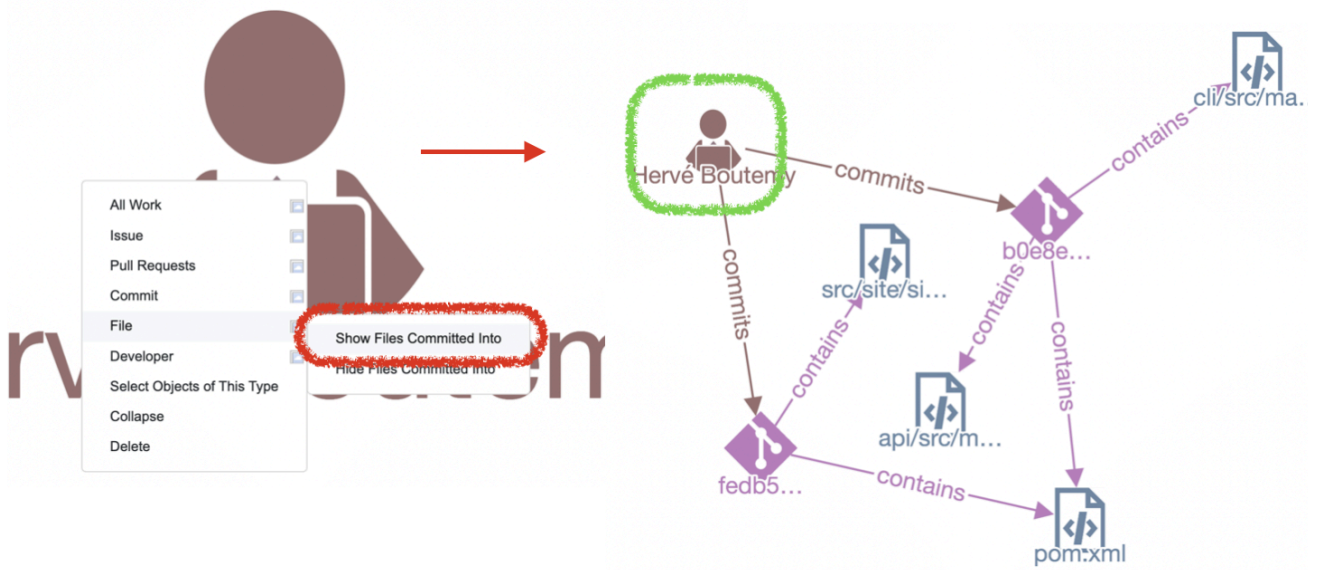


Figure 20: Developer's Context Menu | Show Files Committed Into

File's Context Menu

From the context menu of the file type nodes, the user can learn renamed files, commits, developers that commit into the file, pull requests that modify the file, and related issues about the file.

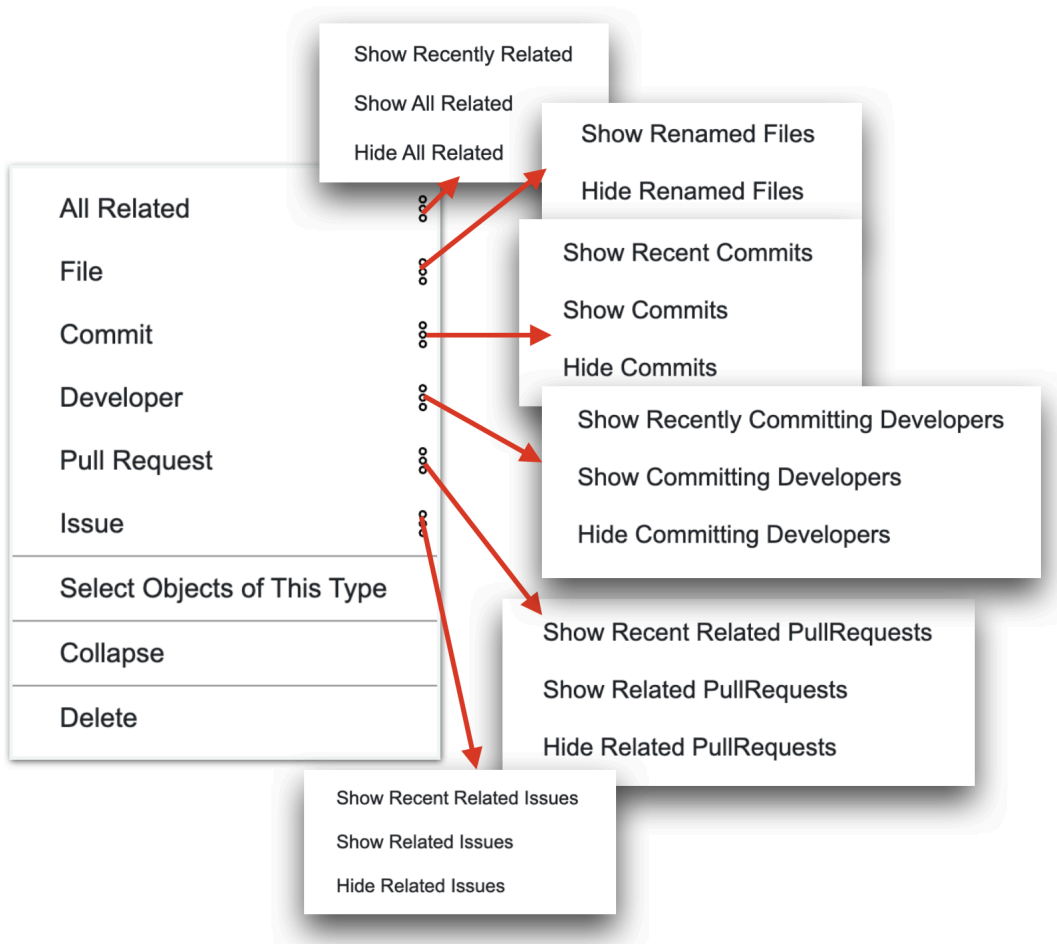


Figure 21: File's Context Menu

Show Issues Related to File: This option gets the issues related to commits that include selected files.

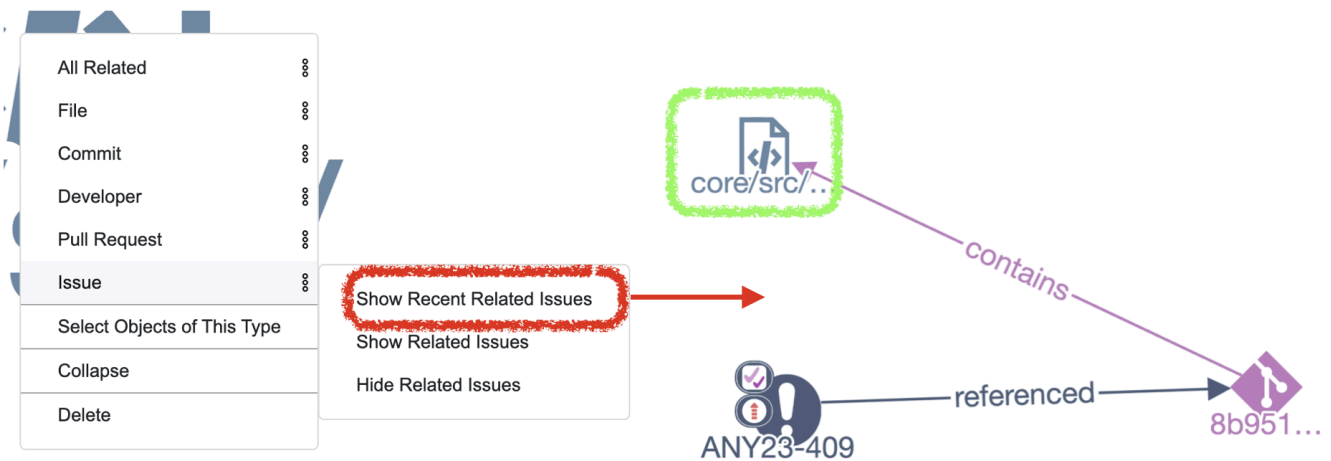


Figure 22: File's Context Menu | Show Related Issues

Issue's Context Menu

From the context menu of the issue-type nodes, the user can learn referenced commits and pull requests, related files, related developers, and developers who commit to the issue.

Figure 23: Issue's Context Menu

Show Developers Commit for Issue: This option gets the developers whose commit references the selected issue.

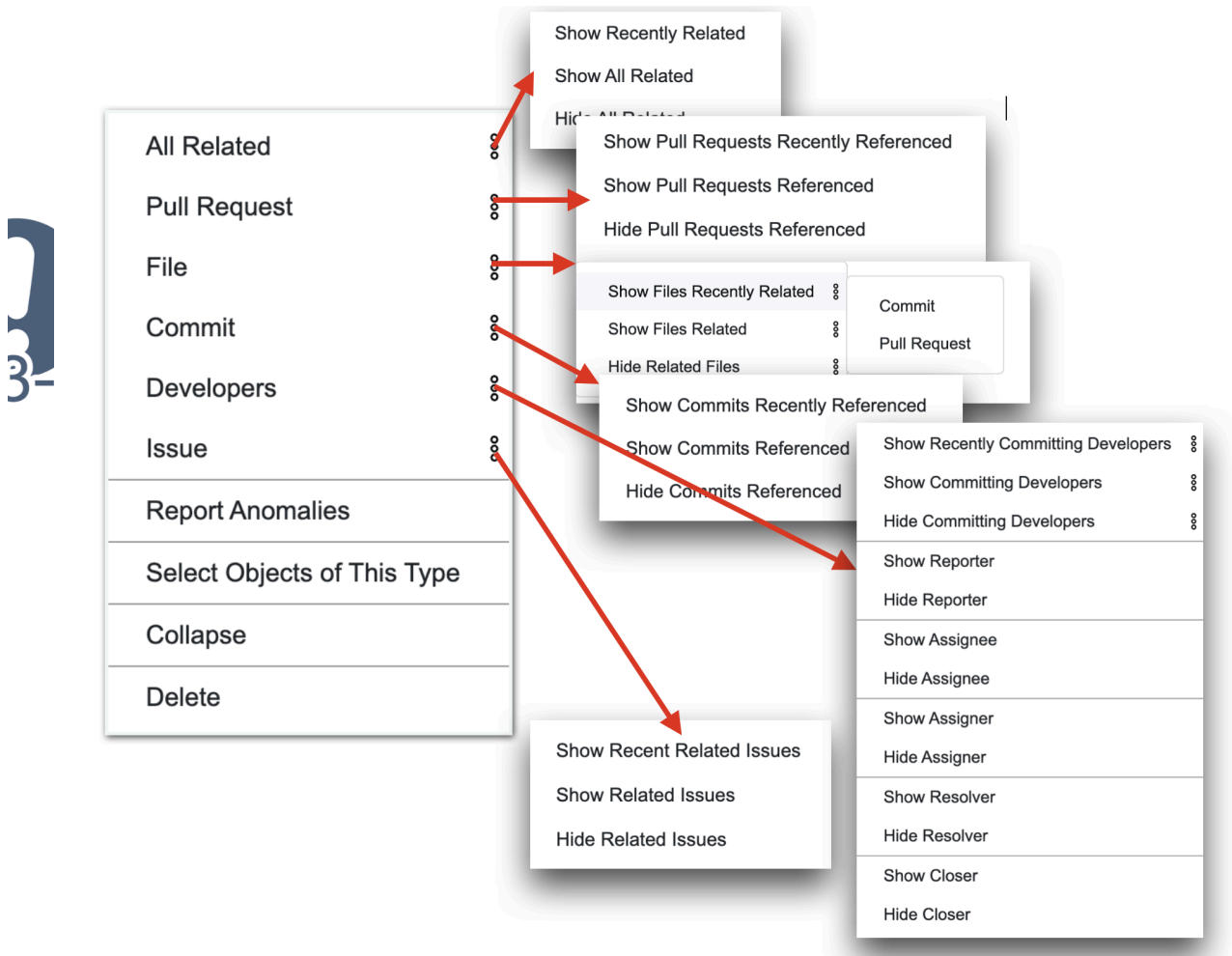


Figure 24: Issue's Context Menu | Show Committed Developers

Pull Request's Context Menu

From the context menu of the pull request type nodes, the user can learn referenced issues, related developers, commits, and files that change with the pull request.

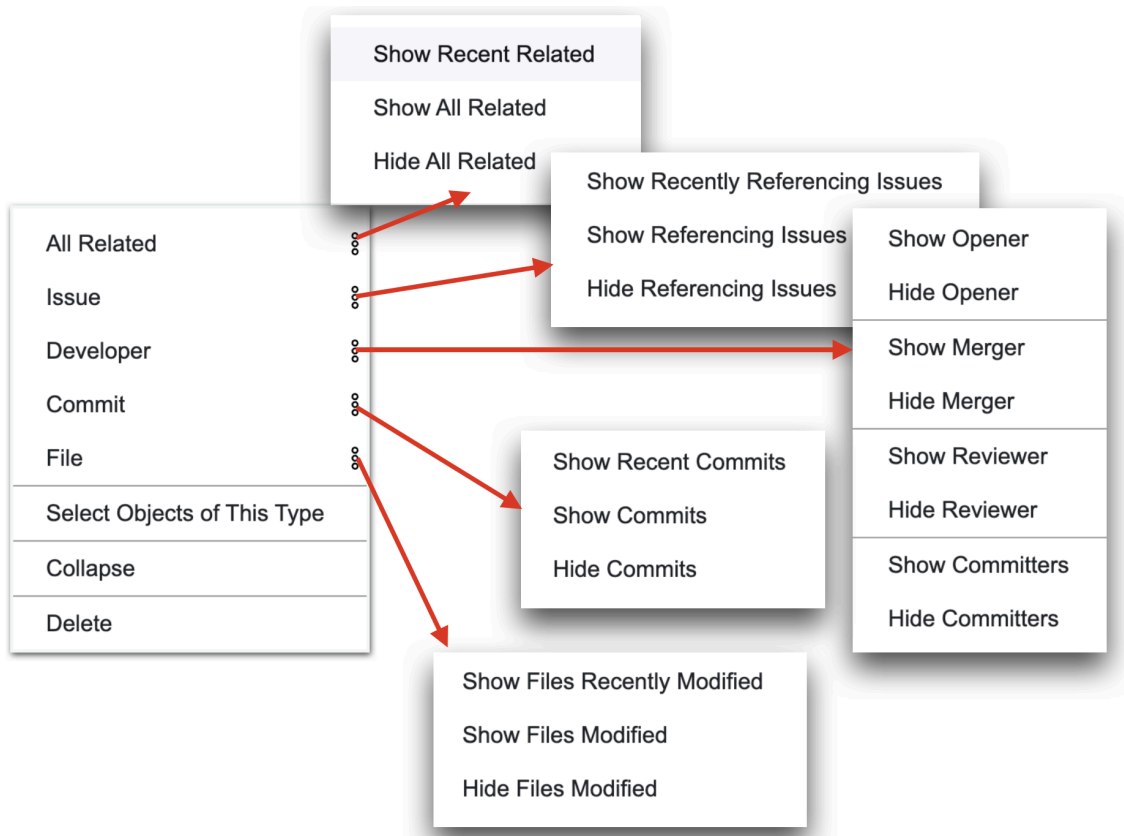


Figure 25: Pull Request Context Menu

Show Changed Files With Pull Request: This option gets the files that change due to the selected pull request.

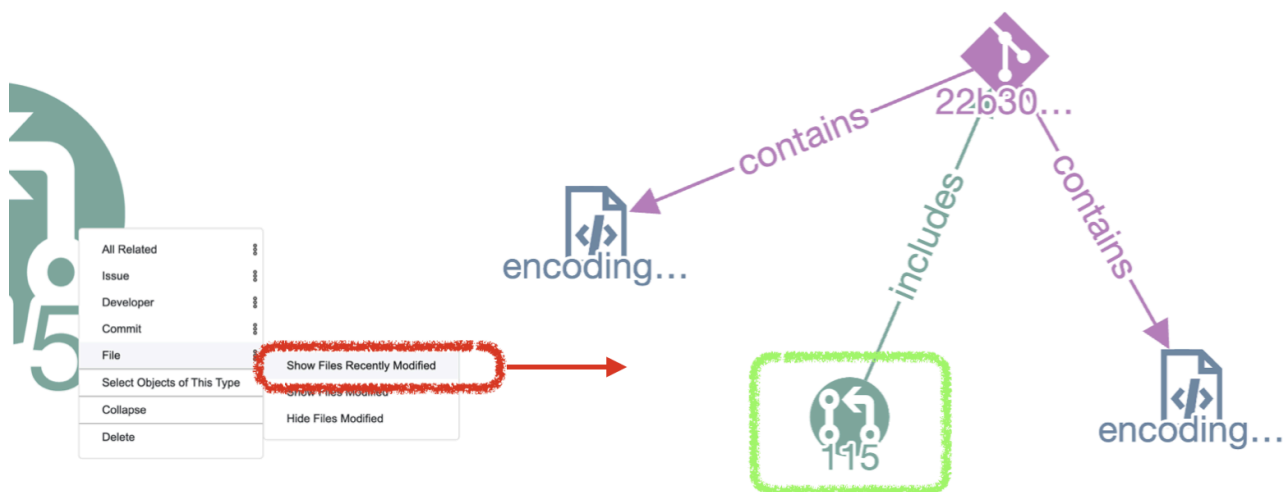


Figure 26: Pull Request Context Menu | Show Files Modified

3 Toolbar

A toolbar is available right under the Menubar to list some frequently needed operations grouped in the same manner as the menu.

Of particular interest is a time range used to limit database queries. When enabled, the time bar will also use this range for the default begin and end values of objects in case the corresponding data does not exist.

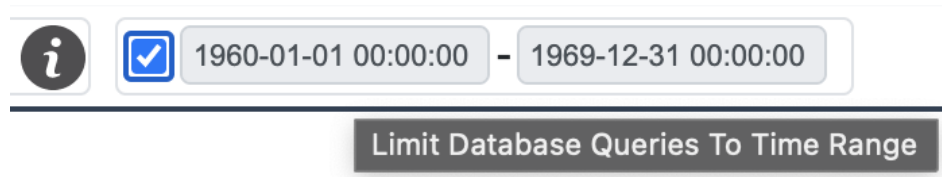


Figure 27: Time range

4 Objects

4.1 Object Inspection Tab

Each node and edge has a set of properties (property-value pairs). If you click on a graph object (a node or an edge) to select it, any current selection will be removed and the graph object that you clicked on will be selected. As a graph object is selected, its properties are shown on the right panel under the Object tab. Below is a map where a pull request was selected and is being inspected in the Object tab in the right panel.

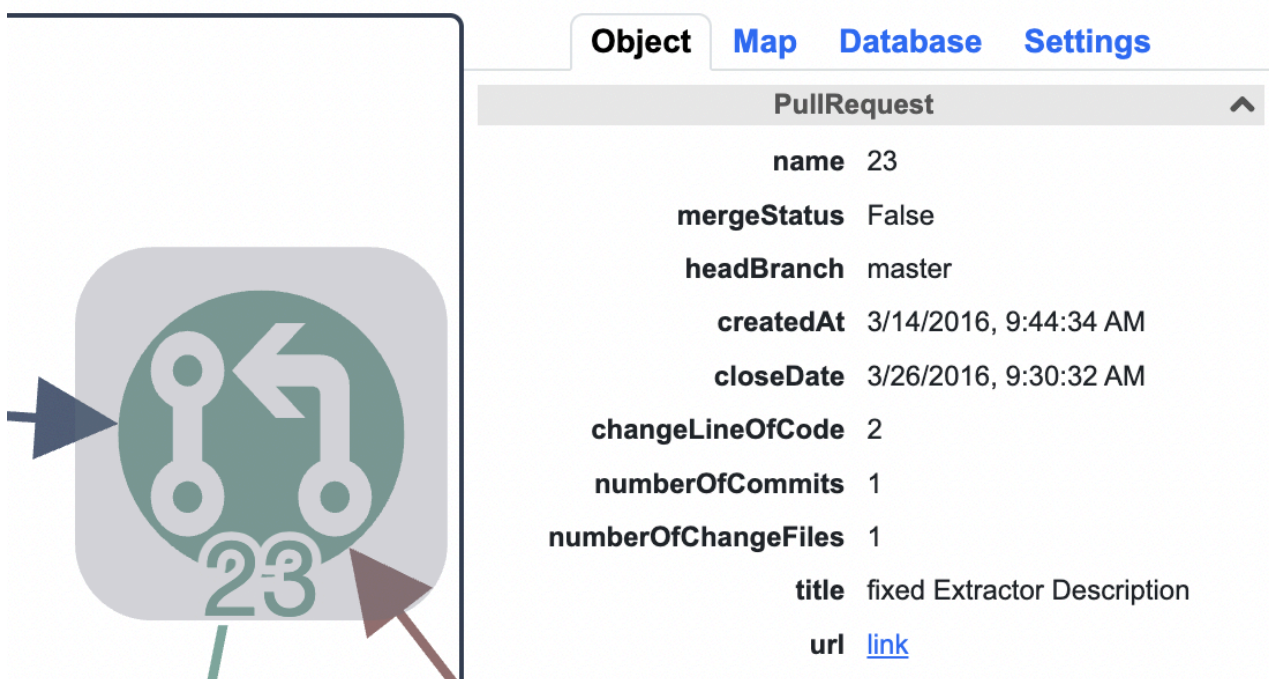


Figure 28: Object Inspection Tab

4.2 Object Statistics Tab

Under the Object tab, there is a sub-section named "Statistics". This section shows statistics about the current objects on the map. An object could be hidden or selected. Statistics will show the count of objects based on their class/type.

#	Type	Count	Selected	Hidden
1	Commit	8	1	1
2	PullRequest	1		
3	Developer	2		1
4	Issue	2	1	
5	INCLUDES	8		1
6	OPENED	1		
7	MERGED	1		1
8	REFERENCED	2		
9	Node	13	2	2
10	Edge	12		2

Figure 29: Object Statistics Tab

4.3 Object Report Tab

The SAA provides users with the ability to report analysis results or observations on Github or Jira platforms if they have authenticated their login to the tool as described in Section [2.1.1](#). Users can submit their report as a comment under a Pull Request in Github or an Issue comment in Jira. This feature allows high integration of SAA with those platforms and aids in usability and practicality. The report tabs are unique to each node type.

4.3.1 Report Issue

In the Report tab, users can seamlessly document their observations to Jira after selecting the issue node. To enhance the clarity of their reports, users have the option to incorporate the current status of the drawing canvas by simply clicking the Graph checkbox. This action adds a rendered image of the graph to the report, providing a visual representation. Additionally, users can directly highlight anomalies detected during their evaluation by checking the "Anomaly" checkbox and specifying the types of anomalies to be considered. For anomaly detection queries you can check Section [6.2.2](#). Once these selections are made, pressing the "Add" button integrates the chosen information into the report. This streamlined process ensures that users can efficiently communicate their findings and contribute to a comprehensive understanding of the issue at hand.

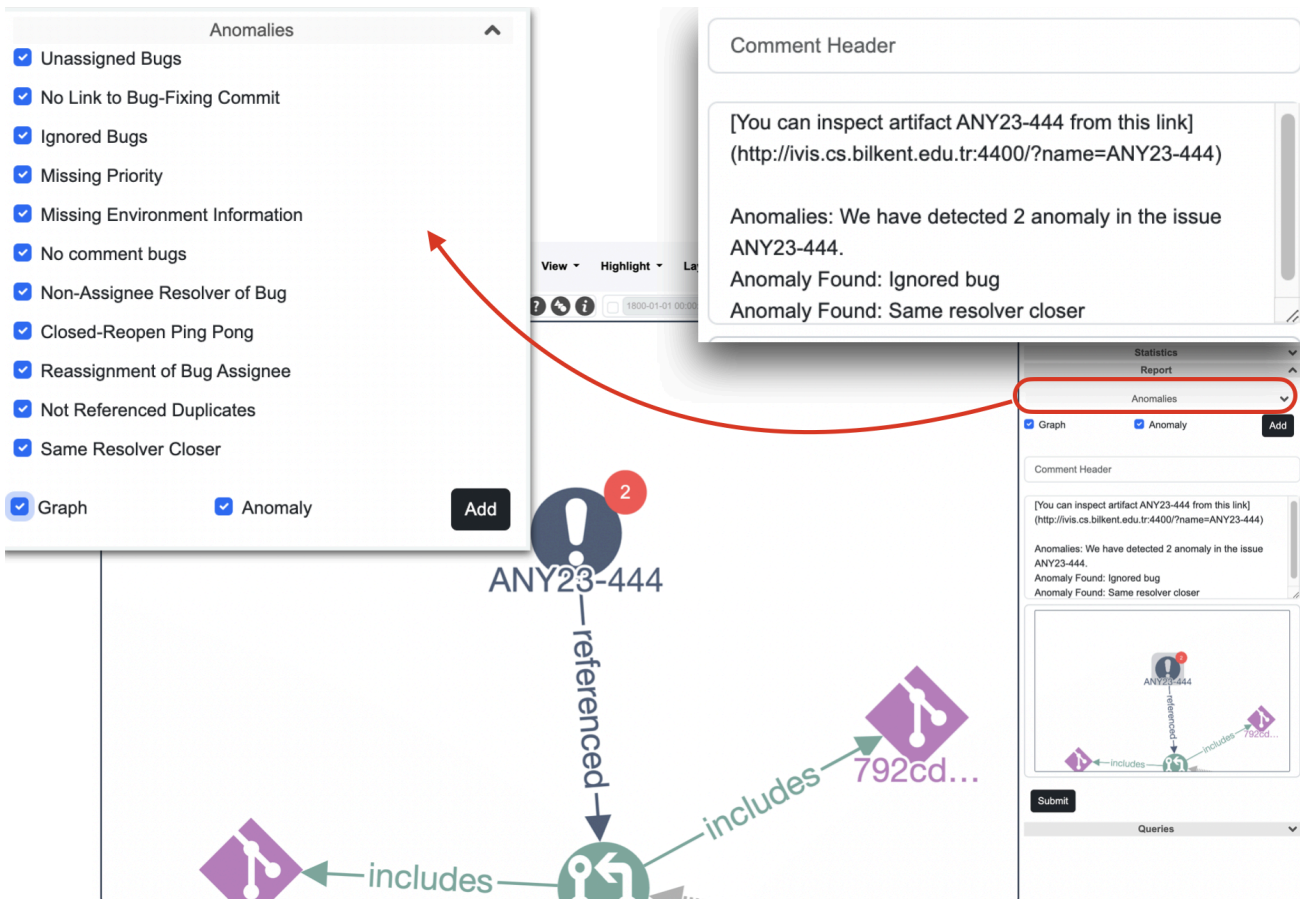


Figure 30: Issue Report Tab

4.3.2 Report Pull Request

In the "Report" tab, users can seamlessly document their observations to Github after selecting the pull request node. To enhance the clarity of their reports, users have the option to incorporate the current status of the drawing canvas by simply clicking the Graph checkbox. This action adds a rendered image of the graph to the report, providing a visual representation. Moreover, users have the option to append reviewer recommendations to the report by marking the "Reviewer Recommendation" checkbox. For details on reviewer recommendations, refer to section 4.4.2. Once these choices are finalized, clicking the "Add" button seamlessly integrates the selected information into the report. This straightforward process ensures users can effectively convey their observations, fostering a thorough understanding of the pull request.

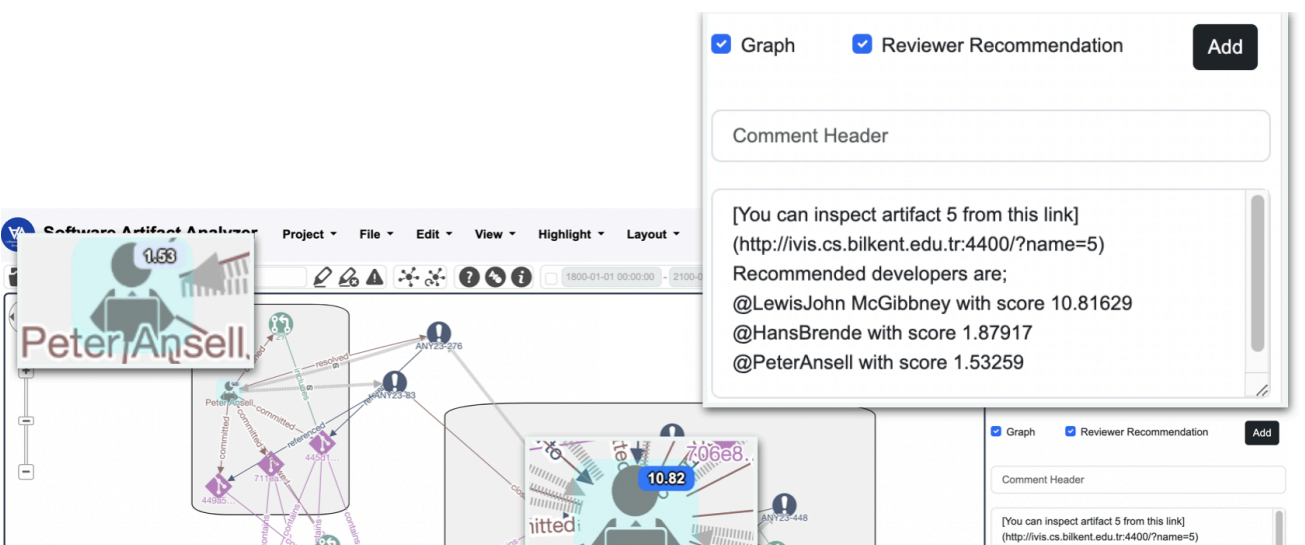


Figure 31: Pull Request Report Tab

4.3.3 Report Commit and File

Within the Commit and File nodes Report tab, users can effortlessly document their observations on GitHub upon selecting the commit node. When reporting commits or files under pull request nodes, users should also specify the pull request node under which they intend to document the commit. To enhance the clarity of their reports, users have the option to incorporate the current status of the drawing canvas by simply clicking the "Graph" checkbox. This action adds a rendered image of the graph to the report, providing a visual representation.

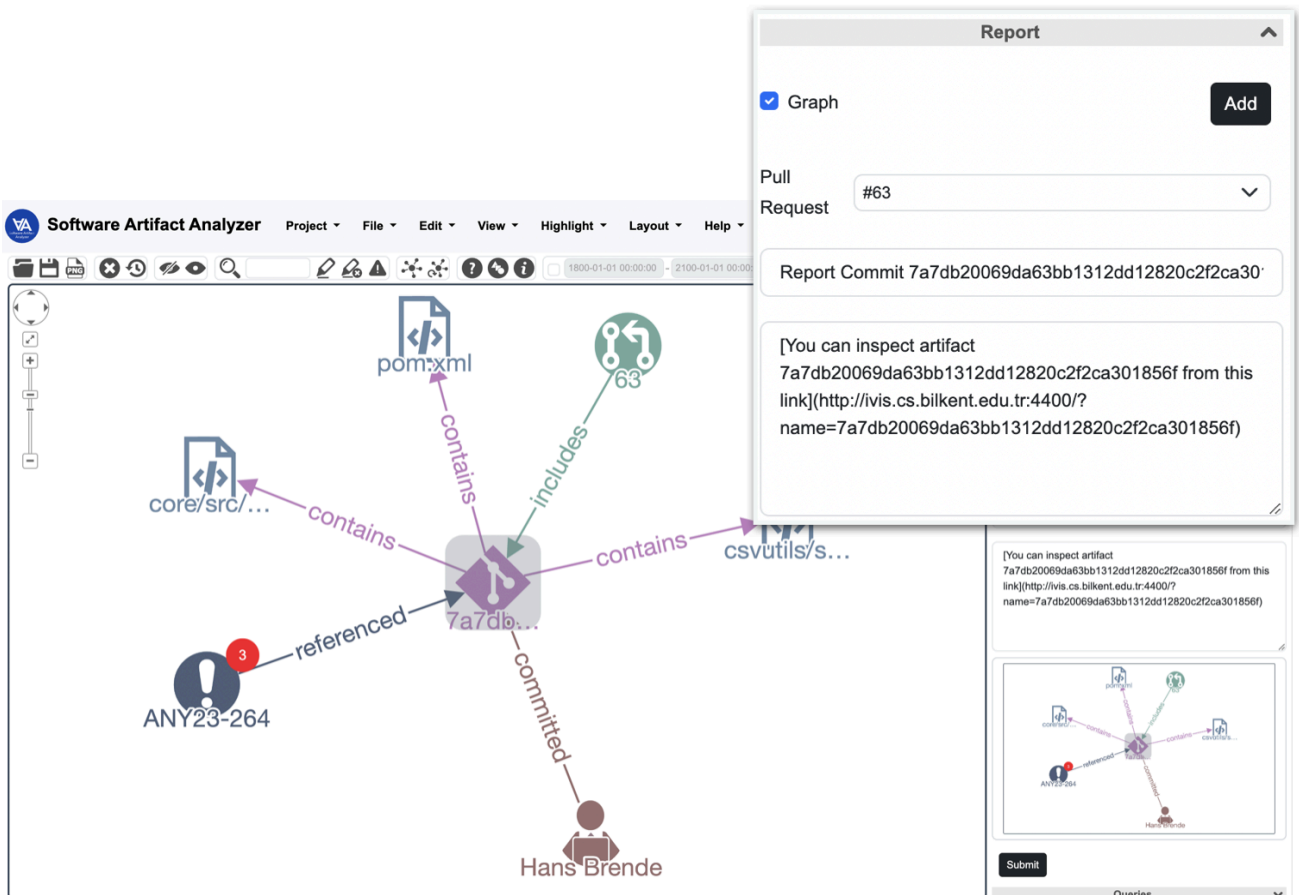


Figure 32: Commit Report Tab

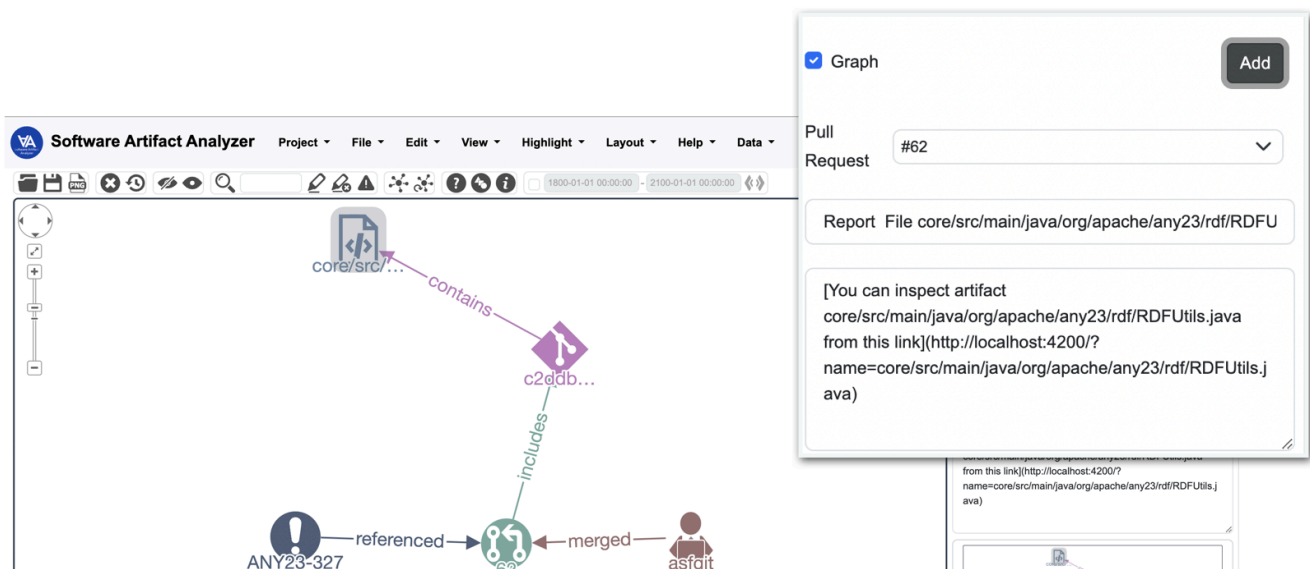


Figure 33: File Report Tab

4.3.5 Report Developer

In the Developer Node Report tab, users can effortlessly document their observations on GitHub or Jira by selecting the developer node. When reporting developer nodes, users should indicate the platform (GitHub or Jira) to which they plan to report and specify the corresponding artifact. For Jira, it should be an issue, and for GitHub, it should be a pull request. To improve the clarity of their reports, users can seamlessly include the current status of the drawing canvas with a simple click on the "Graph" checkbox. This adds a visual representation, and a rendered image of the graph, enhancing the overall comprehensibility of the report.

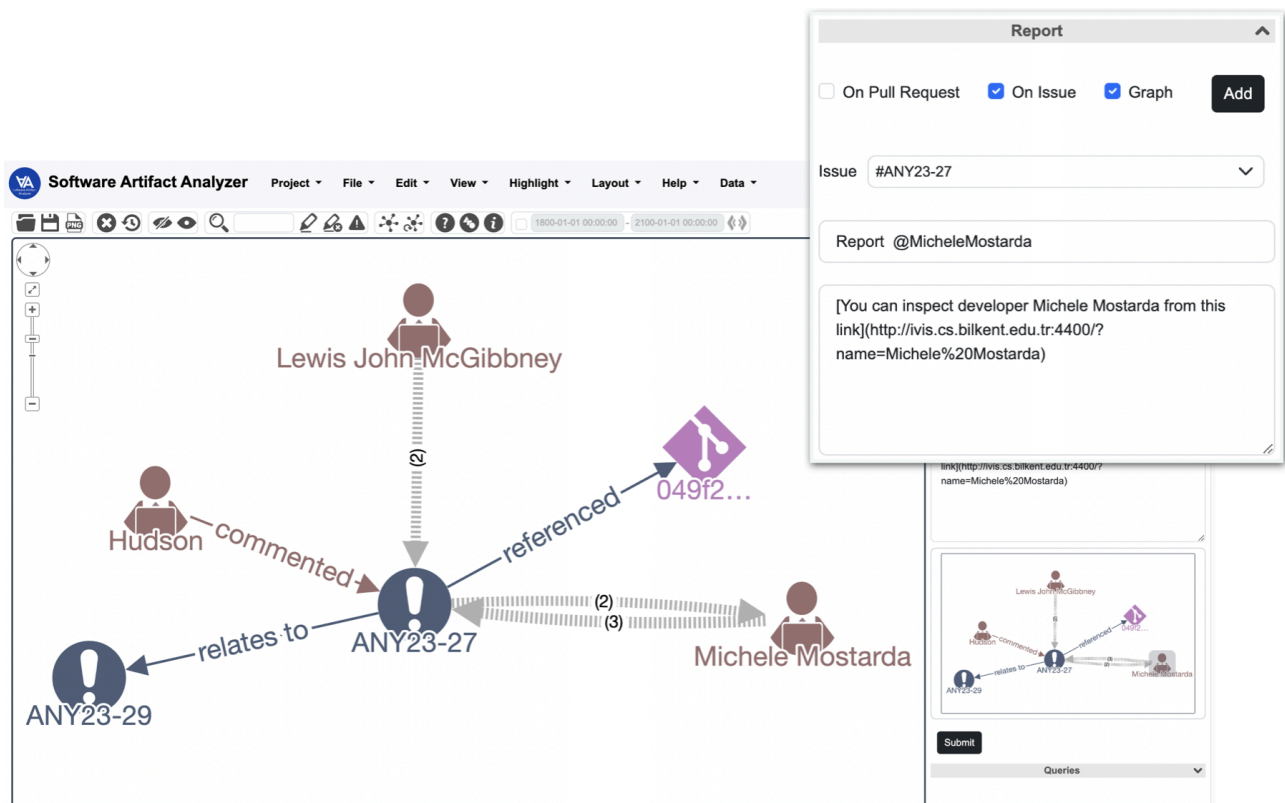


Figure 34: Developer Report Tab

4.4 Object Queries Tab

The queries tab under the object menu is used for analysis of components that are intricately linked to specific node types within the system. These analyses focus on the characteristics, properties, and relationships associated with particular types of nodes. Object queries provide a more targeted approach to analyzing and extracting insights from the data associated with individual node types, ensuring a more fine-grained understanding of the information present in the system. Object queries are specific to each node type.

4.4.1 Expert Recommendation For File Node

In the File Request Query tab users have the option to select the expert recommendation query. The analysis is presented in both table and graph formats, offering flexibility. Users can customize

their view by choosing the number of recommendations to display and incorporating a recency factor for calculation, considering the novelty of the action. The table results furnish a list of recommended reviewers, each accompanied by their respective scores. Meanwhile, the graph results provide the choice to display the graph either clustered by recommended developers or not. Clustering by recommended developers helps users gain a clearer understanding of the relationships between developers and the associated artifacts within the changeset.

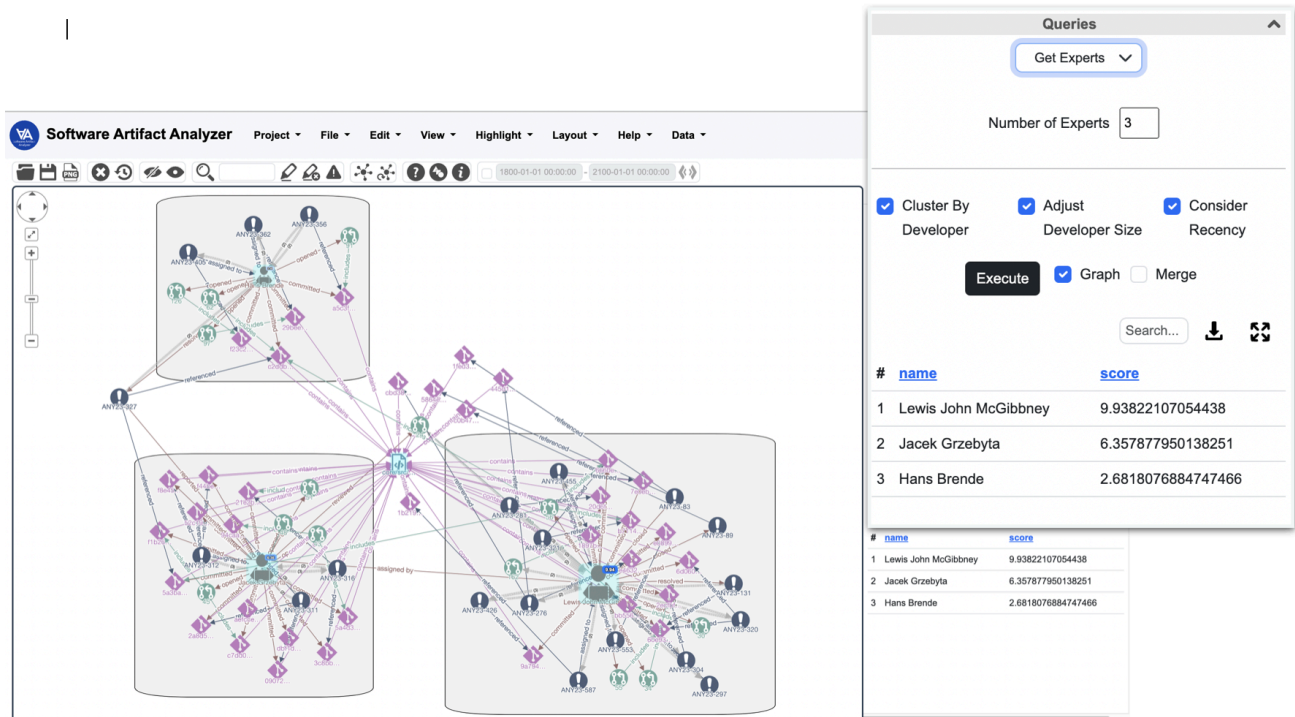


Figure 35: Expert Recommendation Object Query

4.4.2 Reviewer Recommendation For Pull Request Node

In the Pull Request Query tab, users have the option to select the reviewer recommendation query. The analysis is presented in both table and graph formats. Users can customize their view by choosing the number of recommendations to display and incorporating a recency factor for calculation. The table results furnish a list of recommended reviewers, each accompanied by their respective scores. Meanwhile, the graph results provide the choice to display the graph either clustered by recommended developers or not. Clustering by recommended developers helps users gain a clearer understanding of the relationships between developers and the associated artifacts within the changeset.

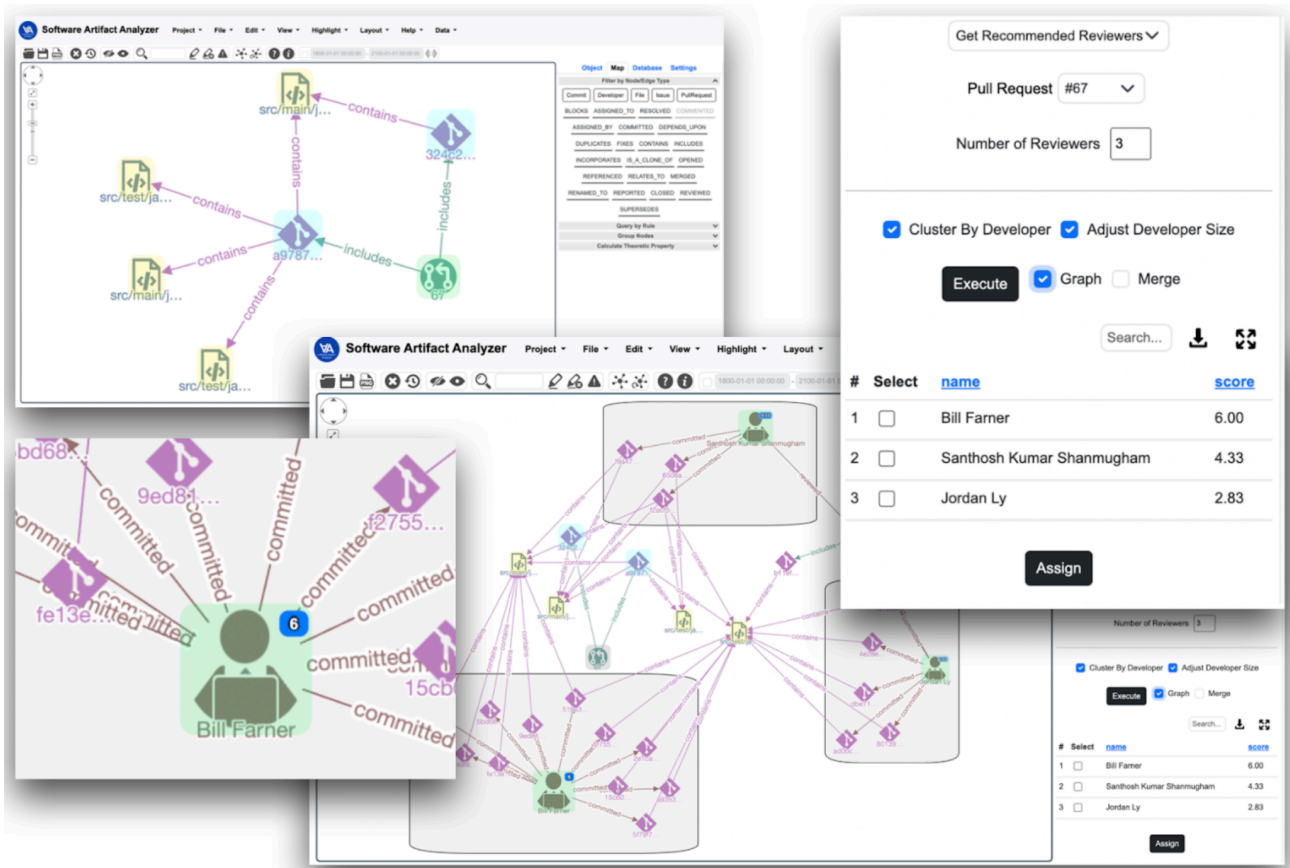


Figure 36: Reviewer Recommendation Object Query

5 Map

5.1 Filter by Node/Edge Type

One simple yet important way to reduce the complexity of a drawing is to filter out certain types of objects or relationships from your map. Visuall [1] facilitates this by providing a button per graph object. Below is an example of a map with all node and edge types (top) and the same map after the COMMENTED edge was filtered out (bottom).

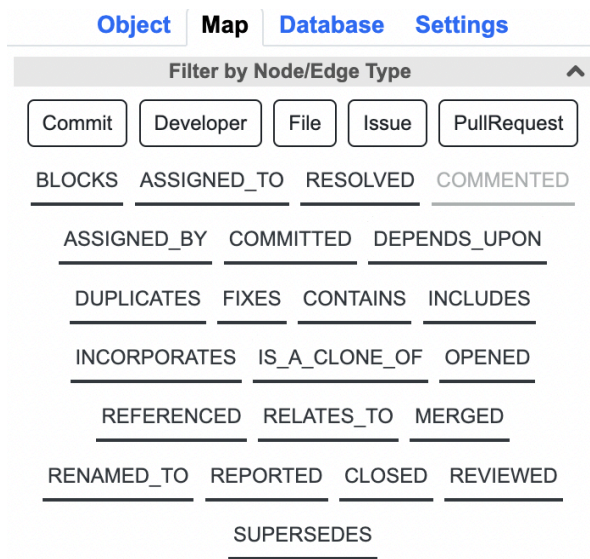


Figure 37: Filter by Node/Edge Type

5.2 Query by Rule

Often, users would like to filter the content available in a database or on the client side using certain rules. SAA, via Visuall [1], enables users to execute their queries. One can call an object with a given property and combine different rules with logic operators. For more detailed information, refer to the [Visuall User Guide](#). For instance, if one is interested in issues with the issue type "Improvement" and created before 2024, they can put together a rule composed of both components as depicted in Figure 28.

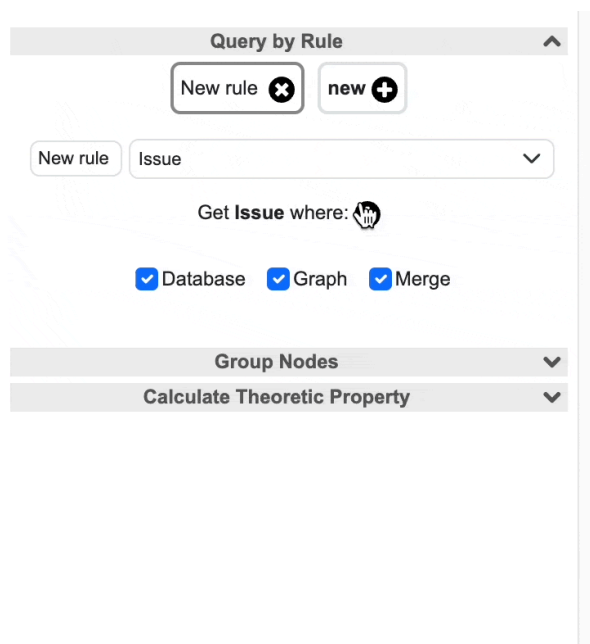


Figure 38: Query by Rule

5.3 Group Nodes

Grouping nodes is a frequently used feature in graph visualization tools to manage graph complexity. Through Visual [1], users can group nodes based on map topology. For more information, one can refer to [Visual User Guide](#). Additionally, users can group nodes based on developers. This means that a developer and the artifacts associated with that developer are grouped. However, if an artifact is associated with multiple developers, it is not grouped with any specific developer.

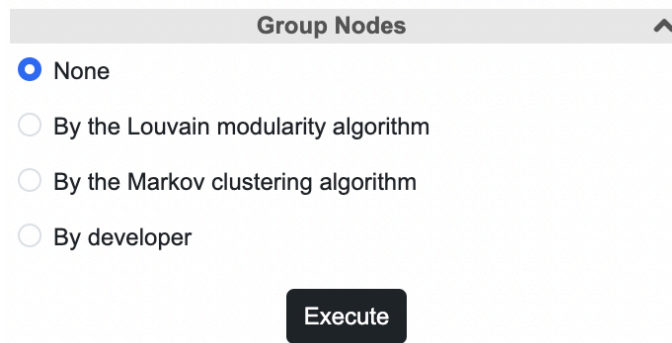


Figure 39: Group Nodes

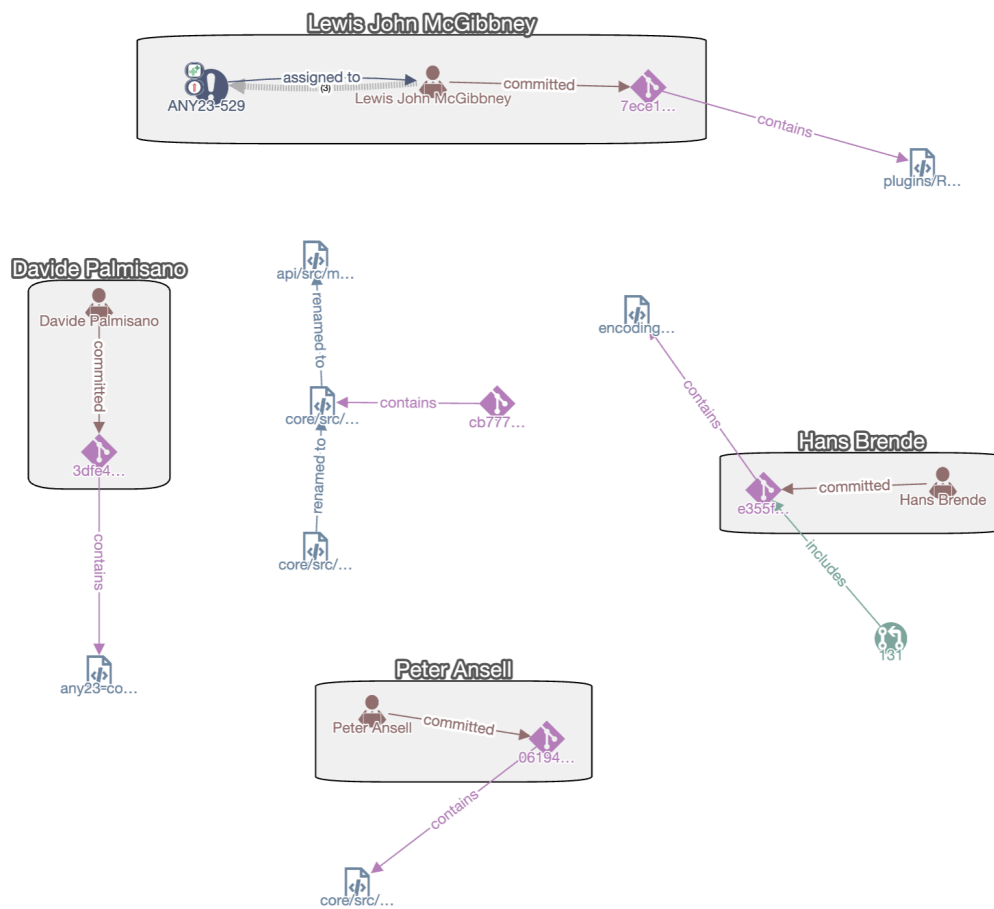


Figure 40: Group by developer

6 Database

6.1 General Queries

Using “General Queries”, the user can execute Neighborhood, Graph of Interest, or Common Targets/Regulators queries. For more details, you can refer to [Visual User Guide](#).

6.2 Custom Queries

The custom queries section is where all queries are stored, including object queries. If you're looking for object-specific queries like reviewer recommendations for pull request nodes, you'll need to indicate the node name. You have the option to choose between text-based results or graphical representations for a more comprehensive analysis, which applies to all queries.

6.2.1 Get Commits of Developer

This is a simple custom object query for getting the commits of the selected developer.

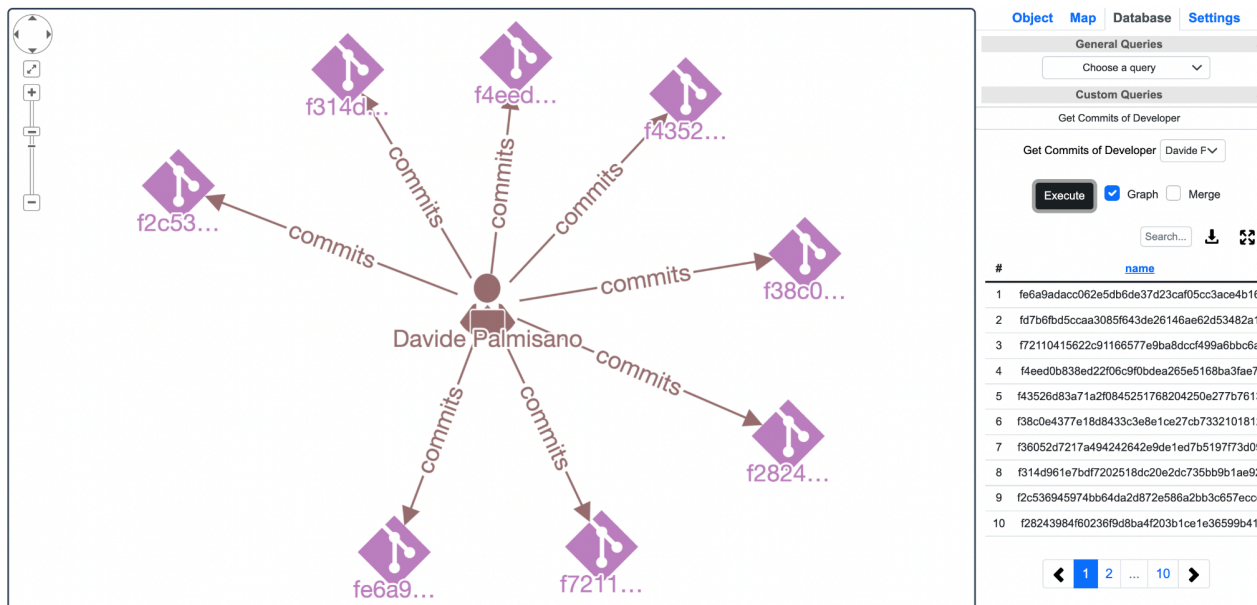


Figure 41: Get Commits of Developer Custom Query

6.2.2 Get Issue Anomalies

During software evaluation, various process anomalies may occur, leading to regression in the process. SAA can detect 11 types of different bug-tracking process anomalies categorized by Qamar et al [2]. A list of the anomalies and their description can be found in the table below. Users can amplify the visual representation of anomalies by incorporating anomaly badges, which showcase the number and types of anomalies detected. These badges are accessible at any point during analysis from the toolbar, as elucidated in [Section 2.4](#).

Anomaly Name	Anomaly Description
Unassigned Bugs	Issues that have been fixed and closed, but not assigned to any person.
No Link to Bug-Fixing Commit	A bug was closed without a commitment link to a bug fix (commit).
Ignored Bugs	Bugs that have been left open for an extended period or have incomplete resolutions.
Missing Priority	A bug that is not prioritized, making it difficult to determine its severity.
Not referenced duplicates	Bugs that are duplicates and have no reference to the original bug.
Missing Environment Information	An error that is missing environment information, such as version, operating system, and product components.
Reassignment of Bug Assignee	The person responsible for resolving the bug has been changed more times than the specified number of times.
No comment bugs	A resolved bug that does not contain a comment.
Non-Assignee Resolver of Bug	The person who resolved the bug is different from the assigned person.
Closed-Reopen Ping Pong	Bugs that are closed and reopened within a specific period.
Same Resolver Closer	The person who resolved the bug is different from the assigned person.

Table 1: Bug process anomalies with descriptions

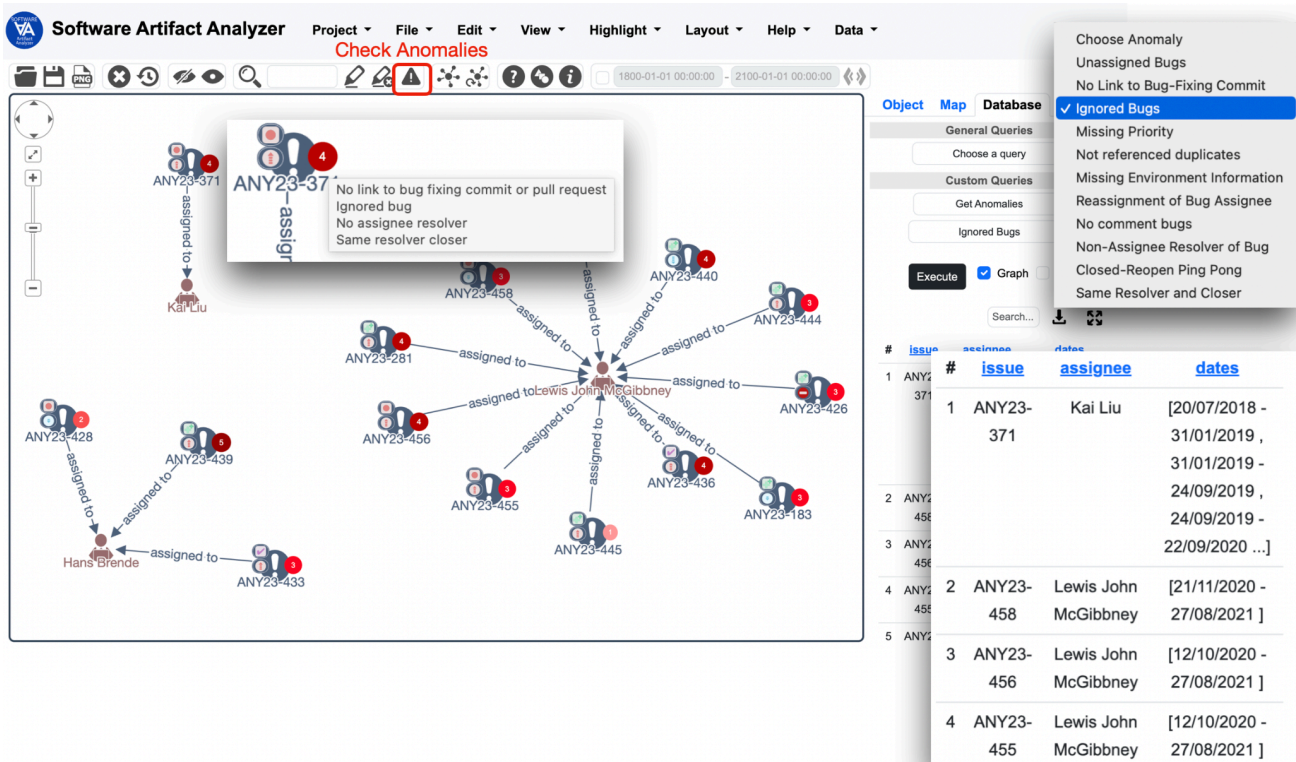


Figure 42: Get Issue Anomalies Custom Query

Furthermore, we acknowledge that specific anomalies, such as 'Ignored Bugs', may necessitate a custom threshold to qualify as anomalies, a requirement that may vary based on project specifications. To address this, we offer customizable settings to adjust their default values accordingly. Within the Settings Anomalies tab, users have the flexibility to define custom thresholds for anomalies, tailoring them to better suit their project needs.

6.2.4 Get Issue Anomaly Statistics

Another handy custom query we have is called "Anomaly Statistics." This query is perfect for gathering issues with a specific number of anomalies. It's handy when you want to focus on learning from issues that contain a high number of anomalies. To run this query, users just need to pick the number of anomalies they're interested in and then click execute. It's a straightforward process that helps users quickly get insights into issues.

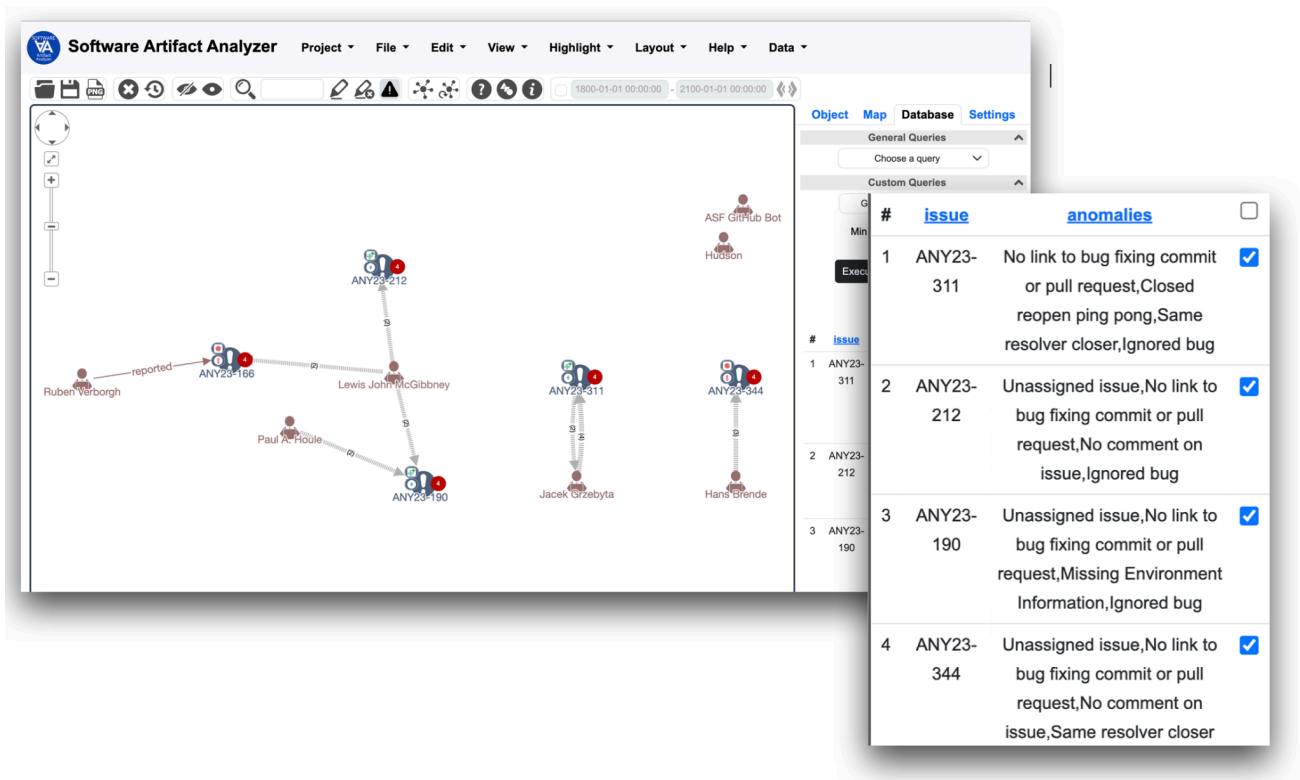


Figure 44: Issue Anomalies Statistics

References

- [1] i-Vis Research Laboratory. 2021. "Visuall: A tool for convenient construction of a web-based visual analysis component", Bilkent University, Ankara, Turkey.
- [2] U. Dogrusoz, M. E. Belviranli, and A. Dilek, 'CiSE: A Circular Spring Embedder Layout Algorithm', IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 6, pp. 953–966, Jun. 2013.
- [3] H. Balci and U. Dogrusoz, "fCoSE: A Fast Compound Graph Layout Algorithm with Constraint Support," in IEEE Transactions on Visualization and Computer Graphics, 28(12), pp. 4582-4593, 2022.
- [4] K. A. Qamar, E. Sülün, and E. Tüzün, "Taxonomy of bug tracking process smells: Perceptions of practitioners and an empirical analysis," Information and Software Technology, vol. 150, p. 106972, 2022. doi:10.1016/j.infsof.2022.106972