# Analysis and Visualization of Unit Test Traces With Kieker and ExplorViz

## 16th Symposium on Software Performance

November 05, 2025

Malte Hansen, David Georg Reichelt, and Wilhelm Hasselbring

C A U
Kiel University
Christian-Albrechts-Universität zu Kiel

ExplorViz

Lancaster University
Leipzig

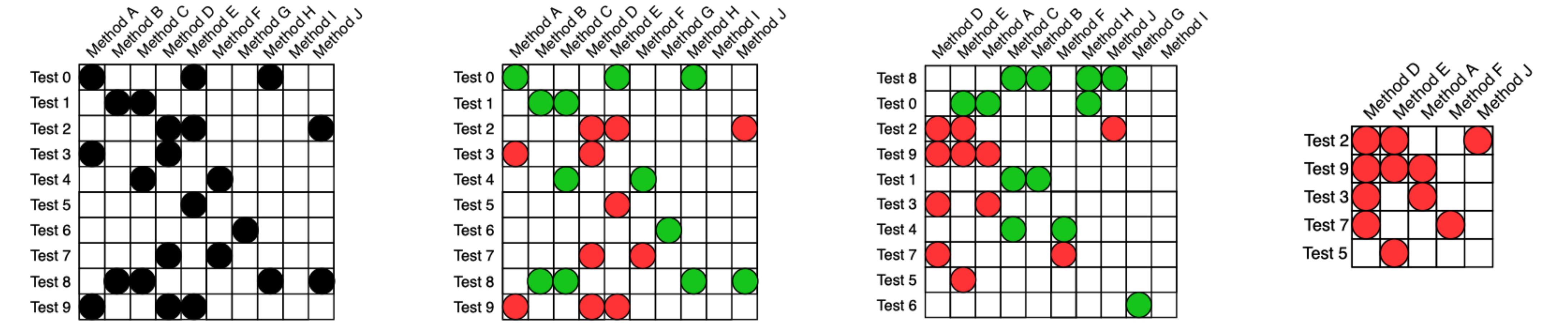
# Motivation

- Software visualization with visual metaphors is a powerful tool to explore different datasets [1]
- Software testing is an integral part of software development

=> Leverage Kieker and ExplorViz to visually explore unit test traces

# Related Work

# Matrix-Based Visualization



|  | Method A | Method B | Method C | Method D | Method E | Method F | Method G | Method H | Method I | Method J |
|---|---|---|---|---|---|---|---|---|---|---|
| Test 0 | ● |  |  |  | ● |  |  | ● |  |  |
| Test 1 |  | ● | ● |  |  |  |  |  |  |  |
| Test 2 |  |  |  | ● | ● |  |  |  |  | ● |
| Test 3 | ● |  |  | ● |  |  |  |  |  |  |
| Test 4 |  |  | ● |  |  | ● |  |  |  |  |
| Test 5 |  |  |  |  | ● |  |  |  |  |  |
| Test 6 |  |  |  |  |  |  | ● |  |  |  |
| Test 7 |  |  |  | ● |  | ● |  |  |  |  |
| Test 8 |  | ● | ● |  |  |  |  | ● |  | ● |
| Test 9 | ● |  |  | ● | ● |  |  |  |  |  |

(a) Test matrix presenting tests and methods as rows and columns; the intersection shows a dot if it was covered.

|  | Method A | Method B | Method C | Method D | Method E | Method F | Method G | Method H | Method I | Method J |
|---|---|---|---|---|---|---|---|---|---|---|
| Test 0 | 🟩 |  |  |  | 🟩 |  |  | 🟩 |  |  |
| Test 1 |  | 🟩 | 🟩 |  |  |  |  |  |  |  |
| Test 2 |  |  |  | 🟥 | 🟥 |  |  |  |  | 🟥 |
| Test 3 | 🟥 |  |  | 🟥 |  |  |  |  |  |  |
| Test 4 |  |  | 🟩 |  |  | 🟩 |  |  |  |  |
| Test 5 |  |  |  |  | 🟥 |  |  |  |  |  |
| Test 6 |  |  |  |  |  |  | 🟩 |  |  |  |
| Test 7 |  |  |  | 🟥 |  | 🟥 |  |  |  |  |
| Test 8 |  | 🟩 | 🟩 |  |  |  |  | 🟩 |  | 🟩 |
| Test 9 | 🟥 |  |  | 🟥 | 🟥 |  |  |  |  |  |

(b) Coverage colored according to pass (green) or fail (red).

|  | Method D | Method E | Method A | Method C | Method B | Method F | Method H | Method J | Method G | Method I |
|---|---|---|---|---|---|---|---|---|---|---|
| Test 8 |  |  |  | 🟩 | 🟩 |  | 🟩 | 🟩 |  |  |
| Test 0 |  | 🟩 | 🟩 |  |  |  | 🟩 |  |  |  |
| Test 2 | 🟥 | 🟥 |  |  |  |  |  | 🟥 |  |  |
| Test 9 | 🟥 | 🟥 | 🟥 |  |  |  |  |  |  |  |
| Test 1 |  |  |  | 🟩 | 🟩 |  |  |  |  |  |
| Test 3 | 🟥 |  | 🟥 |  |  |  |  |  |  |  |
| Test 4 |  |  |  | 🟩 |  | 🟩 |  |  |  |  |
| Test 7 | 🟥 |  |  |  |  | 🟥 |  |  |  |  |
| Test 5 |  | 🟥 |  |  |  |  |  |  |  |  |
| Test 6 |  |  |  |  |  |  |  |  | 🟩 |  |

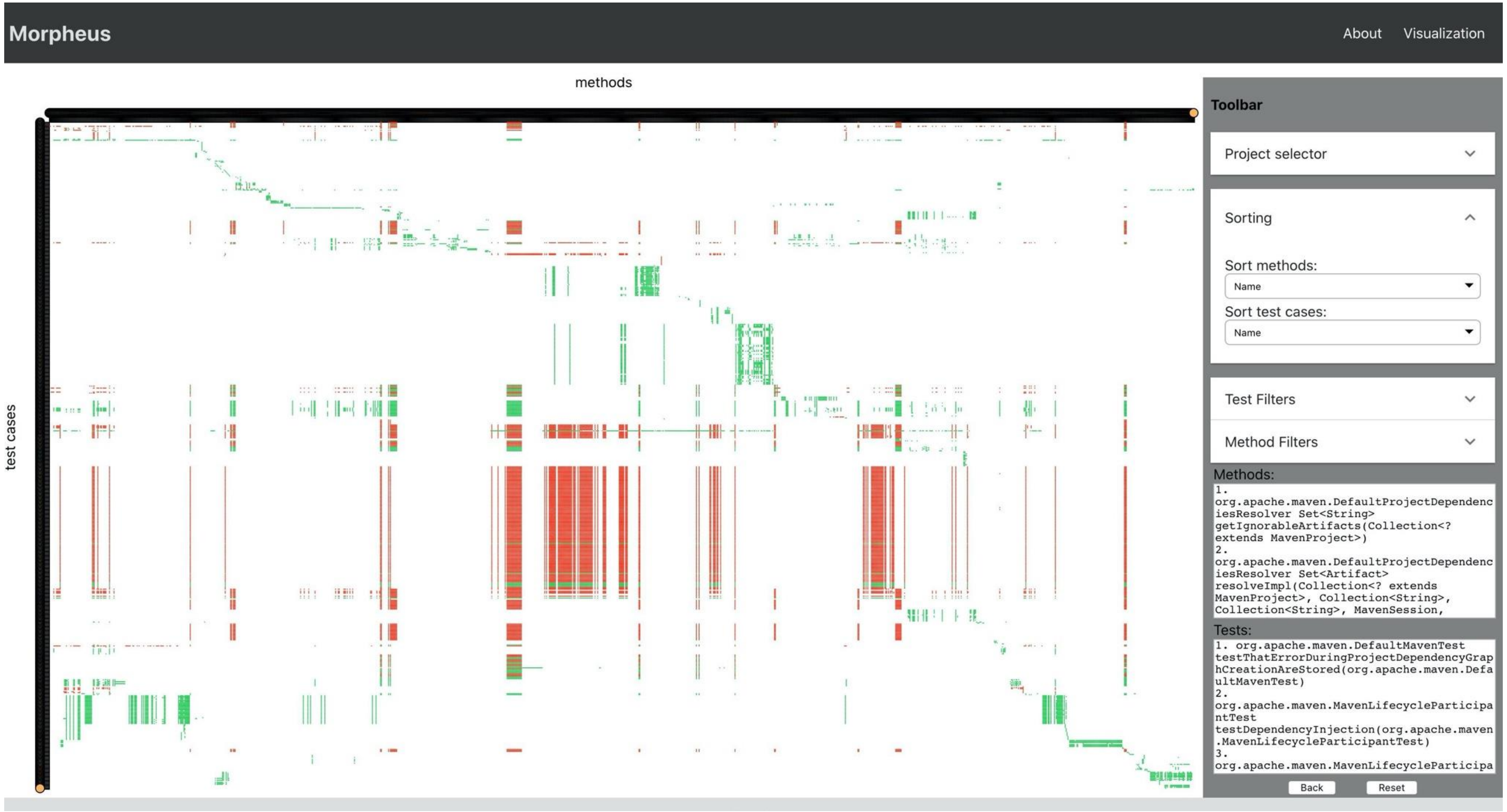
(c) Sorting tests and methods by coverage.

|  | Method D | Method E | Method A | Method F | Method J |
|---|---|---|---|---|---|
| Test 2 | 🟥 | 🟥 |  |  | 🟥 |
| Test 9 | 🟥 | 🟥 | 🟥 |  |  |
| Test 3 | 🟥 |  | 🟥 |  |  |
| Test 7 | 🟥 |  |  | 🟥 |  |
| Test 5 |  | 🟥 |  |  |  |

(d) Filtering to only failing tests and the methods executed by them.

[2]

# Matrix Visualization by Dreef et al.



Methods:

```
1.
org.apache.maven.DefaultProjectDependenc
iesResolver Set<String>
getIgnorableArtifacts(Collection<?
extends MavenProject>)
2.
org.apache.maven.DefaultProjectDependenc
iesResolver Set<Artifact>
resolveImpl(Collection<? extends
MavenProject>, Collection<String>,
Collection<String>, MavenSession,
```

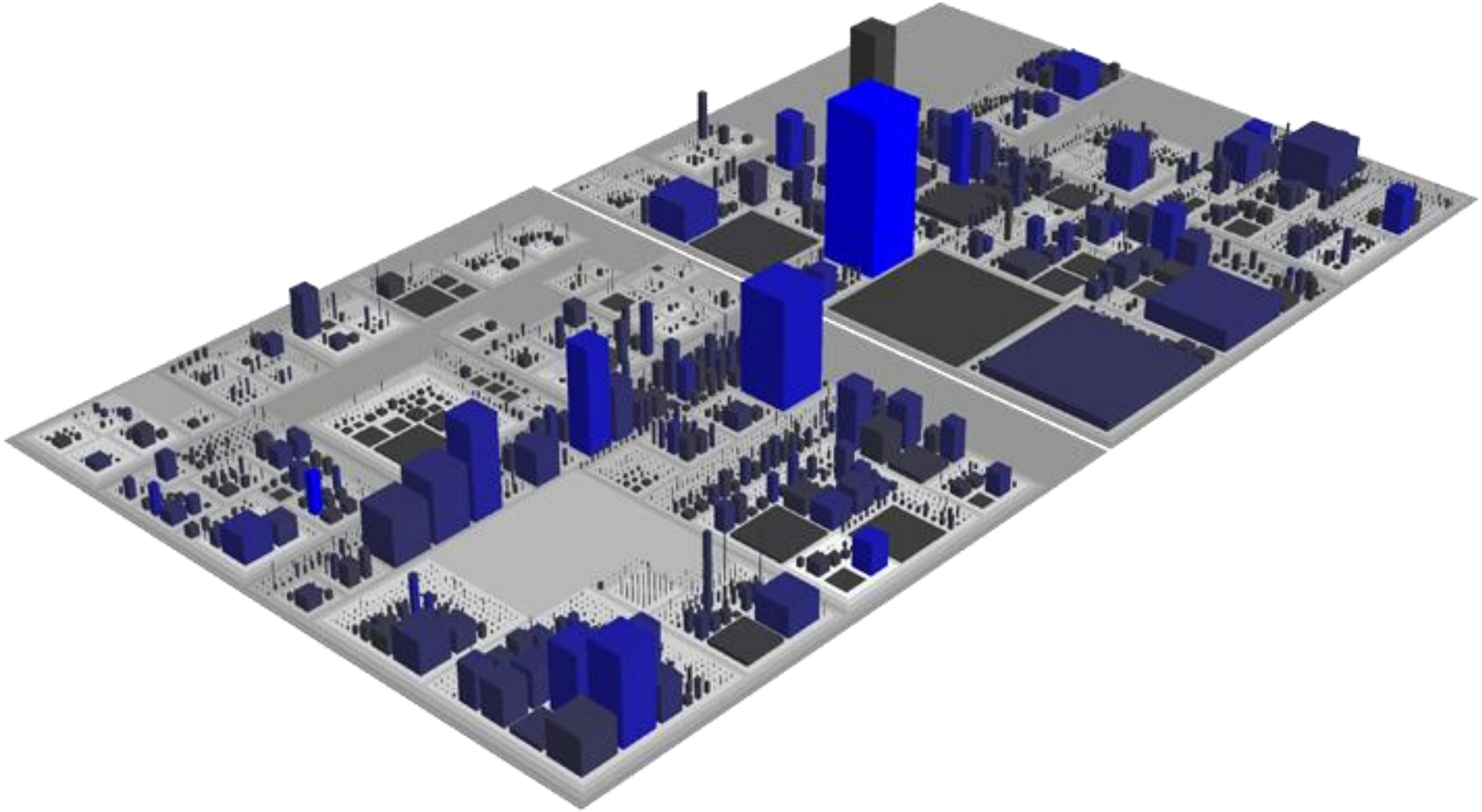
Tests:

```
1. org.apache.maven.DefaultMavenTest
testThatErrorDuringProjectDependencyGrap
hCreationAreStored(org.apache.maven.Defa
ultMavenTest)
2.
org.apache.maven.MavenLifecycleParticipa
ntTest
testDependencyInjection(org.apache.maven
.MavenLifecycleParticipantTest)
3.
org.apache.maven.MavenLifecycleParticipa
```

[2]

=> Missing relation to software's architecture and evolution

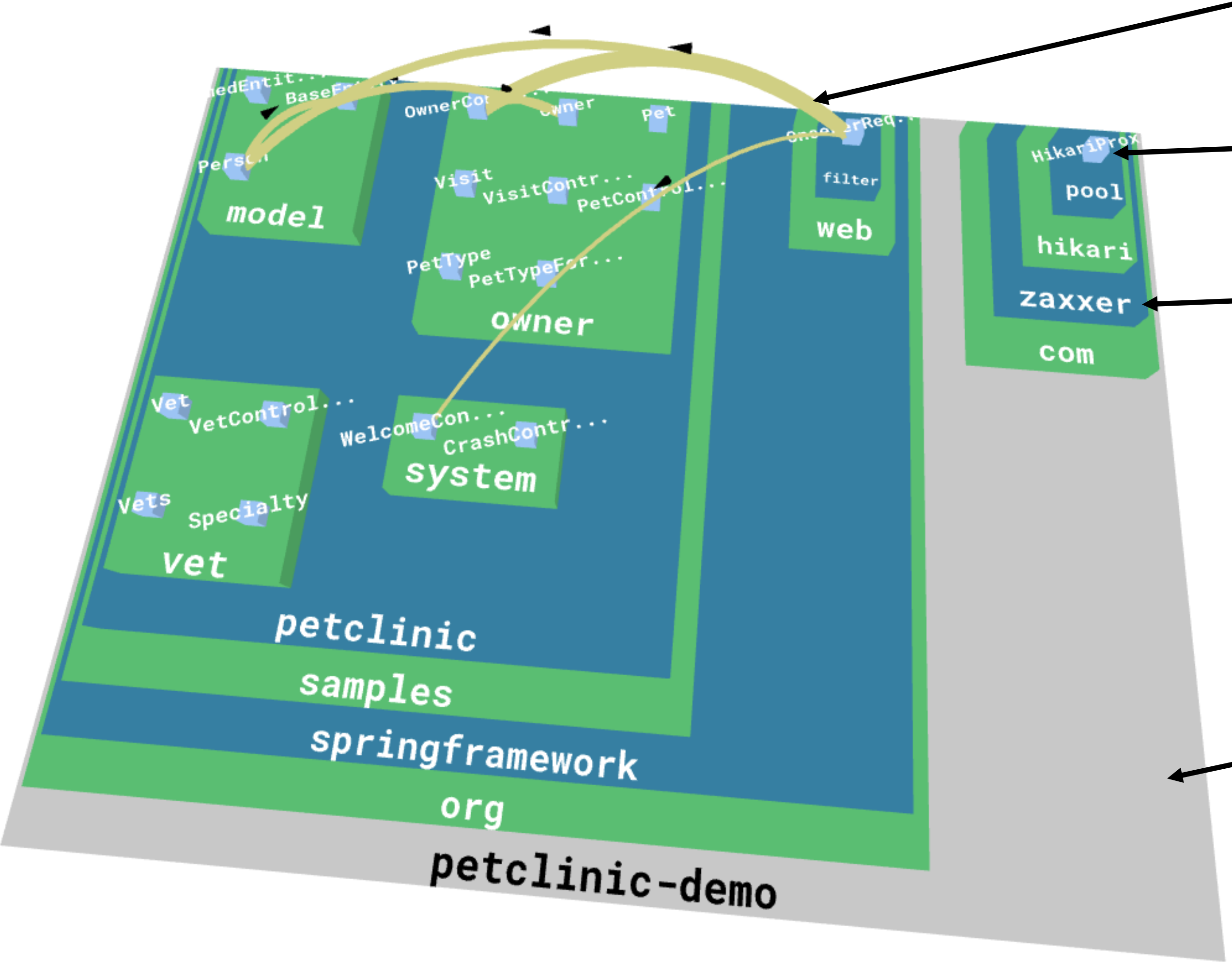
# CodeCity



[3]

# Background

# ExplorViz Visualization



Communication

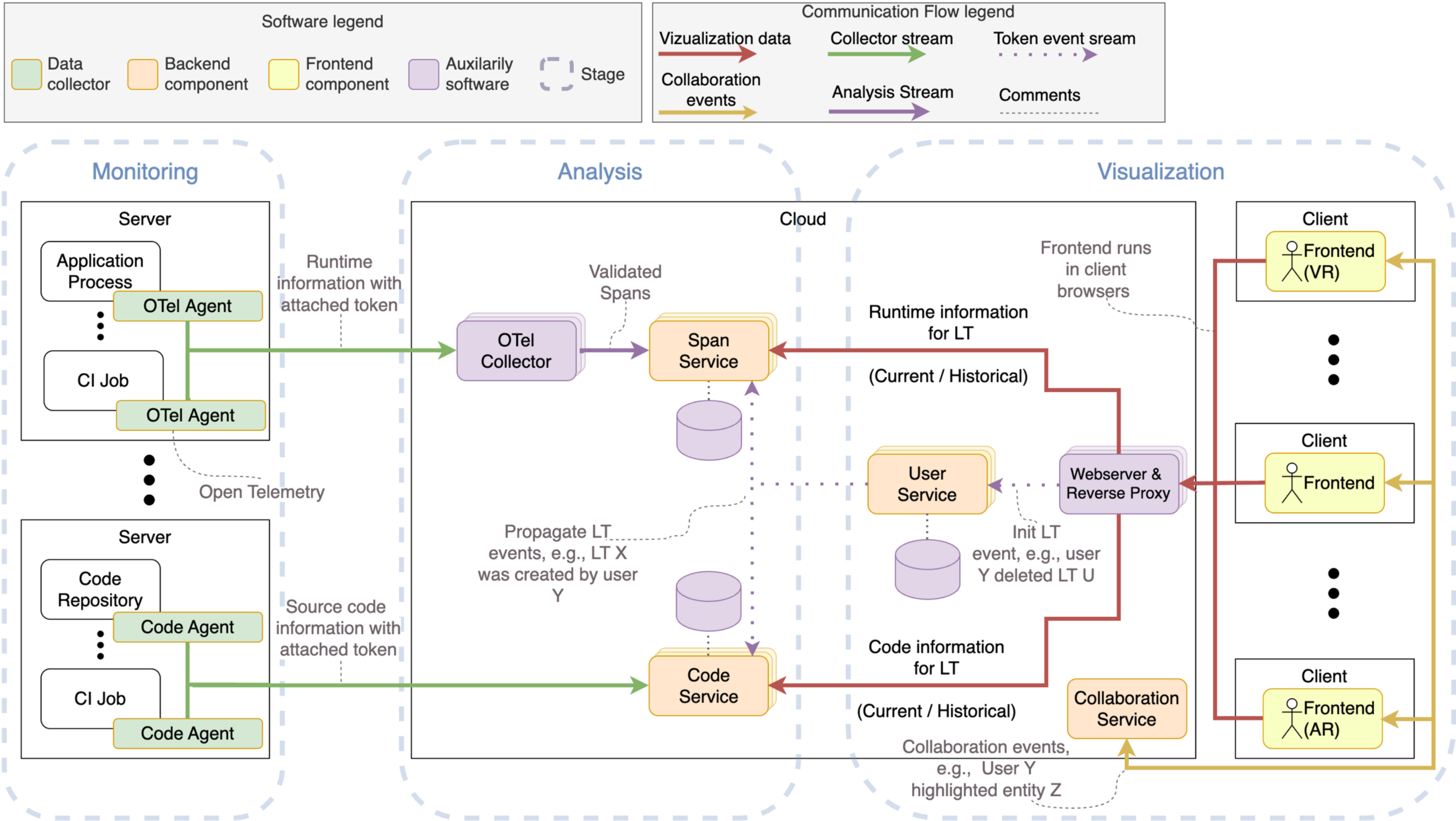
Class

Package

Application

[4]

# ExplorViz Architecture



Software legend

Data collector

Backend component

Frontend component

Auxilarily software

Stage

Communication Flow legend

Vizualization data

Collector stream

Token event sream

Collaboration events

Analysis Stream

Comments

Monitoring

Analysis

Visualization

Server

Application Process

OTel Agent

CI Job

OTel Agent

Open Telemetry

Server

Code Repository

Code Agent

CI Job

Code Agent

Runtime information with attached token

Source code information with attached token

Cloud

OTel Collector

Validated Spans

Span Service

Propagate LT events, e.g., LT X was created by user Y

Code Service

Runtime information for LT

(Current / Historical)

User Service

Webserver & Reverse Proxy

Init LT event, e.g., user Y deleted LT U

Code information for LT

(Current / Historical)

Collaboration Service

Collaboration events, e.g., User Y highlighted entity Z

Frontend runs in client browsers

Client

Frontend (VR)

Client

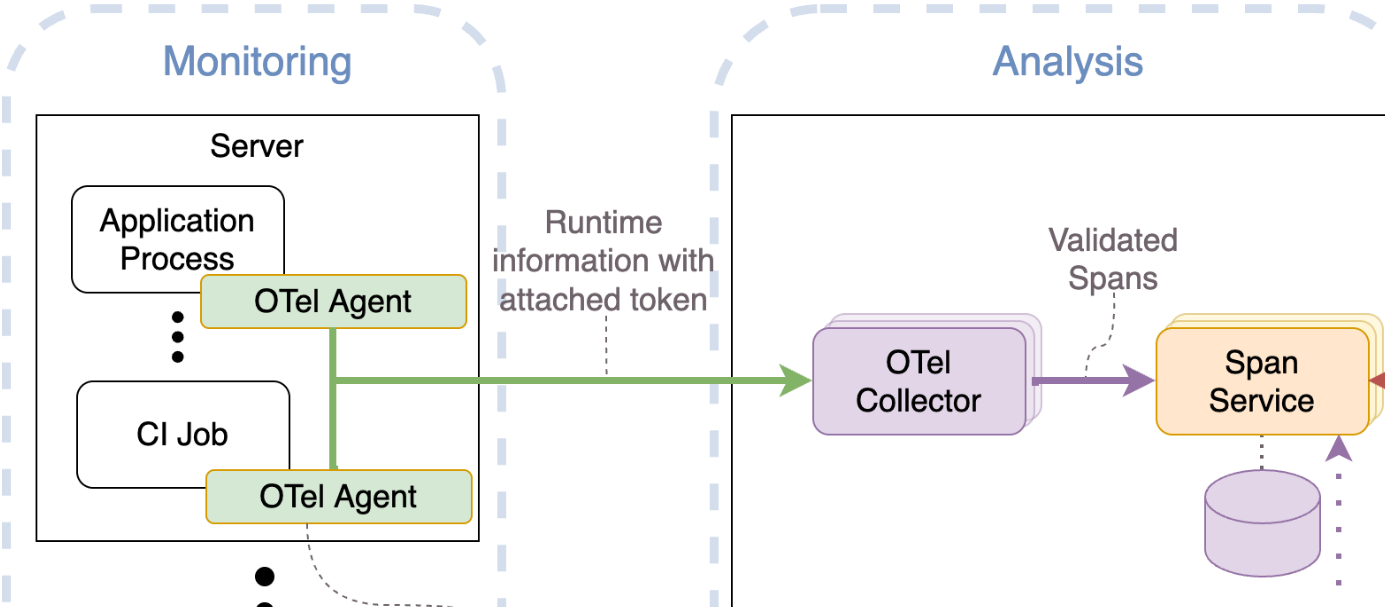
Frontend

Client

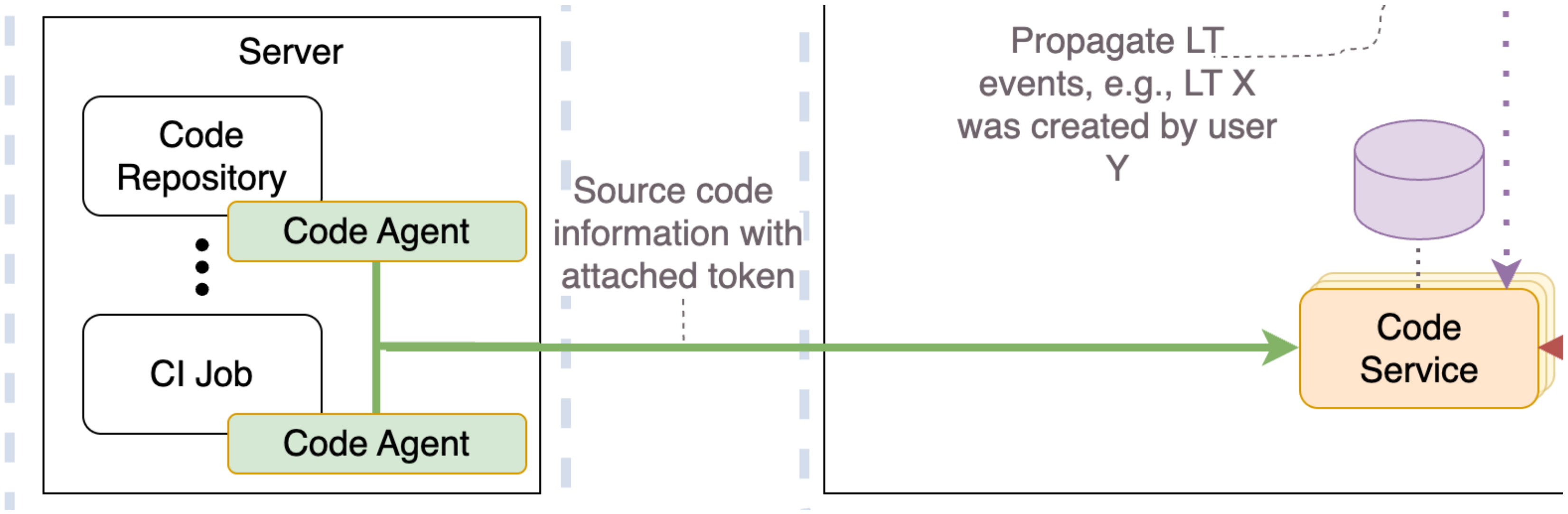
Frontend (AR)

https://explorviz.uni-kiel.de

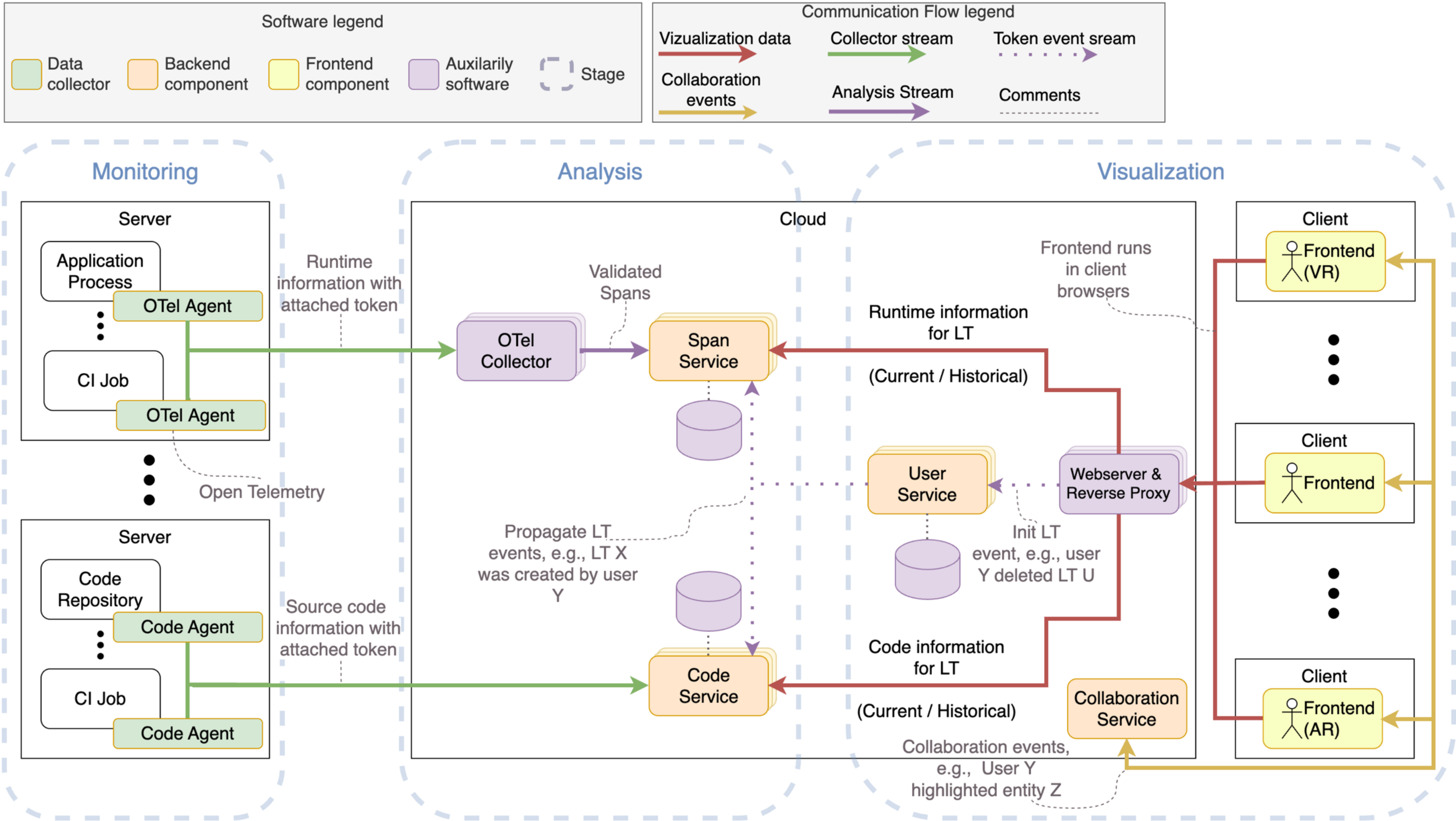
# ExplorViz Architecture



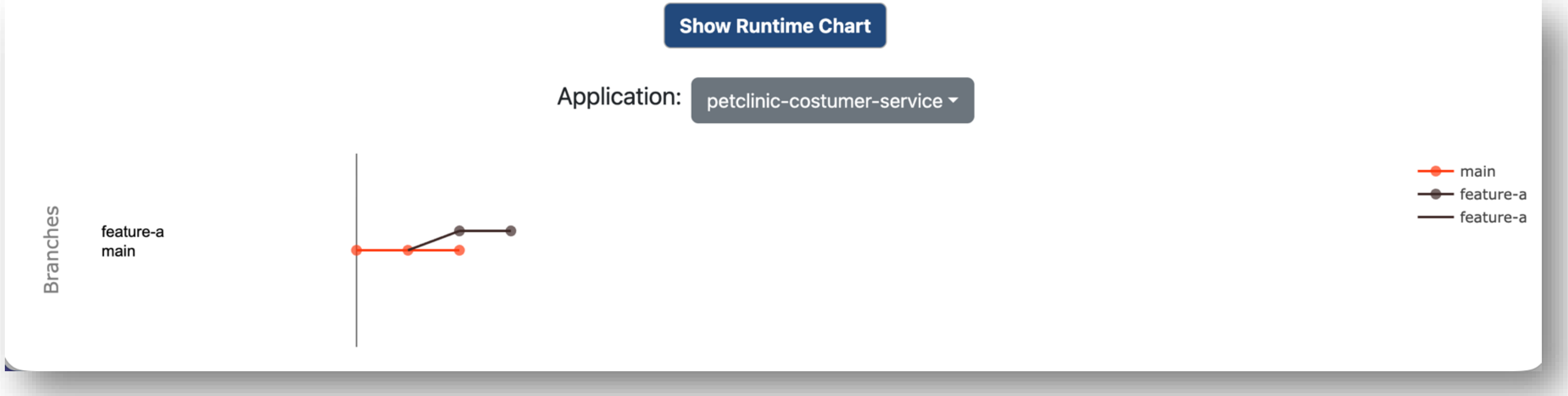
# ExplorViz Architecture


Server
Code Repository
Code Agent
CI Job
Code Agent
Source code information with attached token
Propagate LT events, e.g., LT X was created by user Y
Code Service

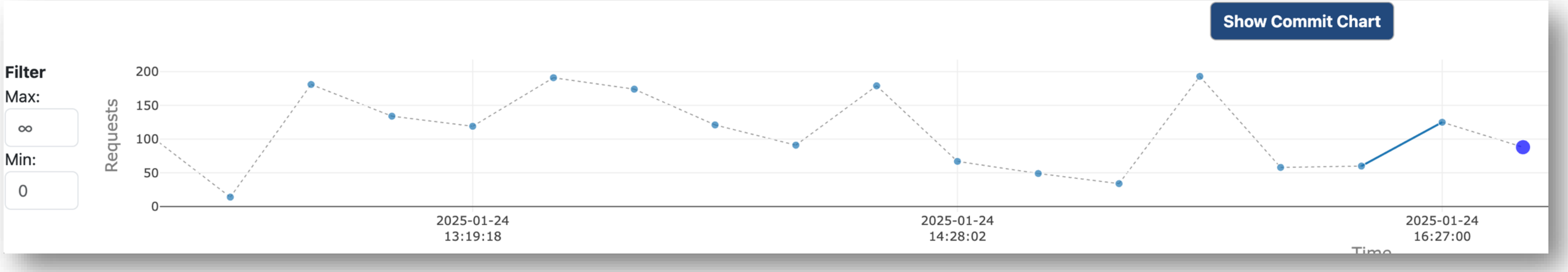
# ExplorViz Architecture



Commit Tree:

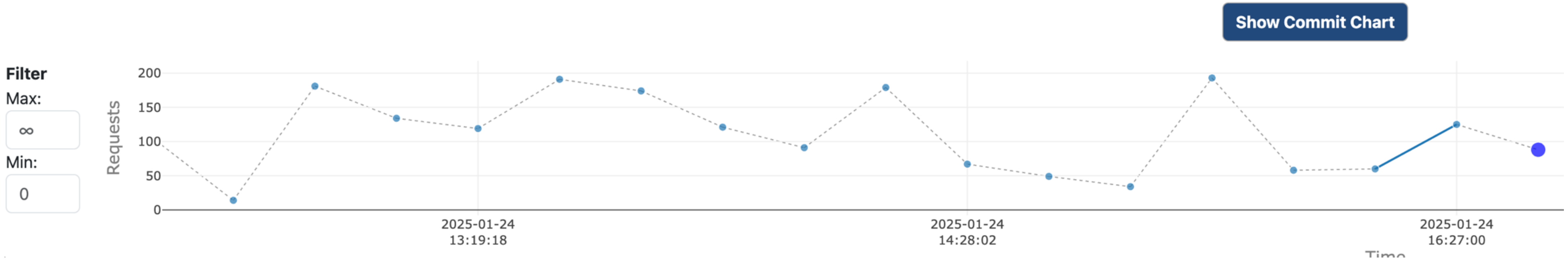


Timeline:



ExplorViz

# Timeline for Trace Data

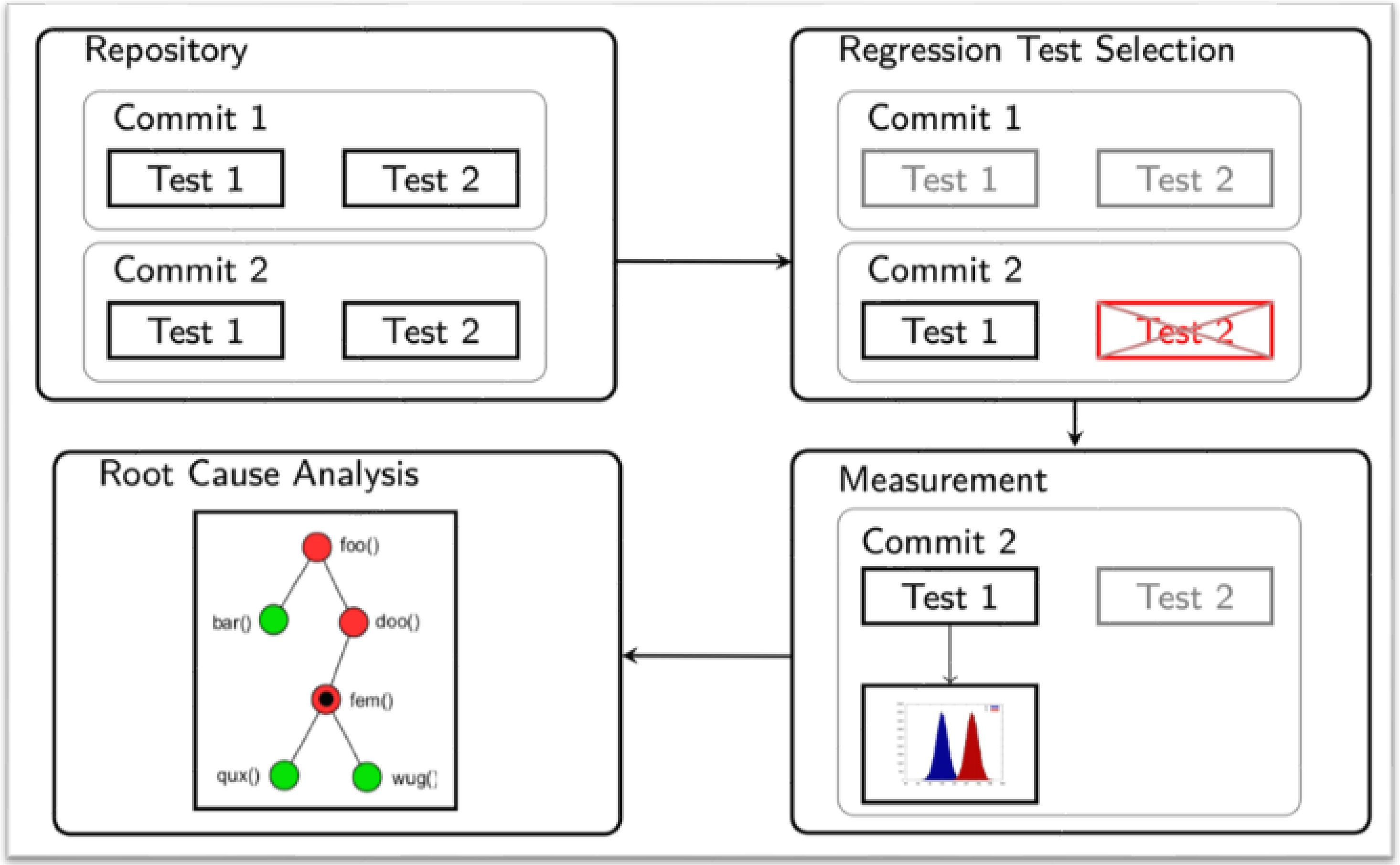


Filter

Max: ∞

Min: 0

# Peass (Performance analysis of software systems)



=> Used to automatically instrument unit tests with Kieker [5]

# Approach

# Trace Gathering Process

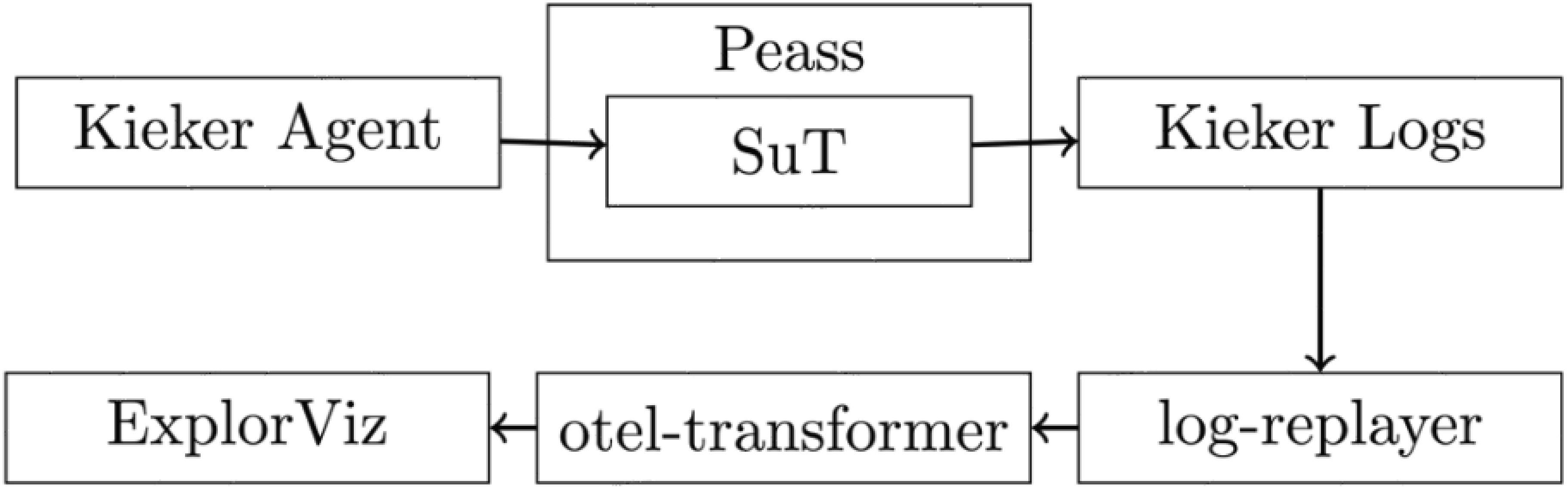
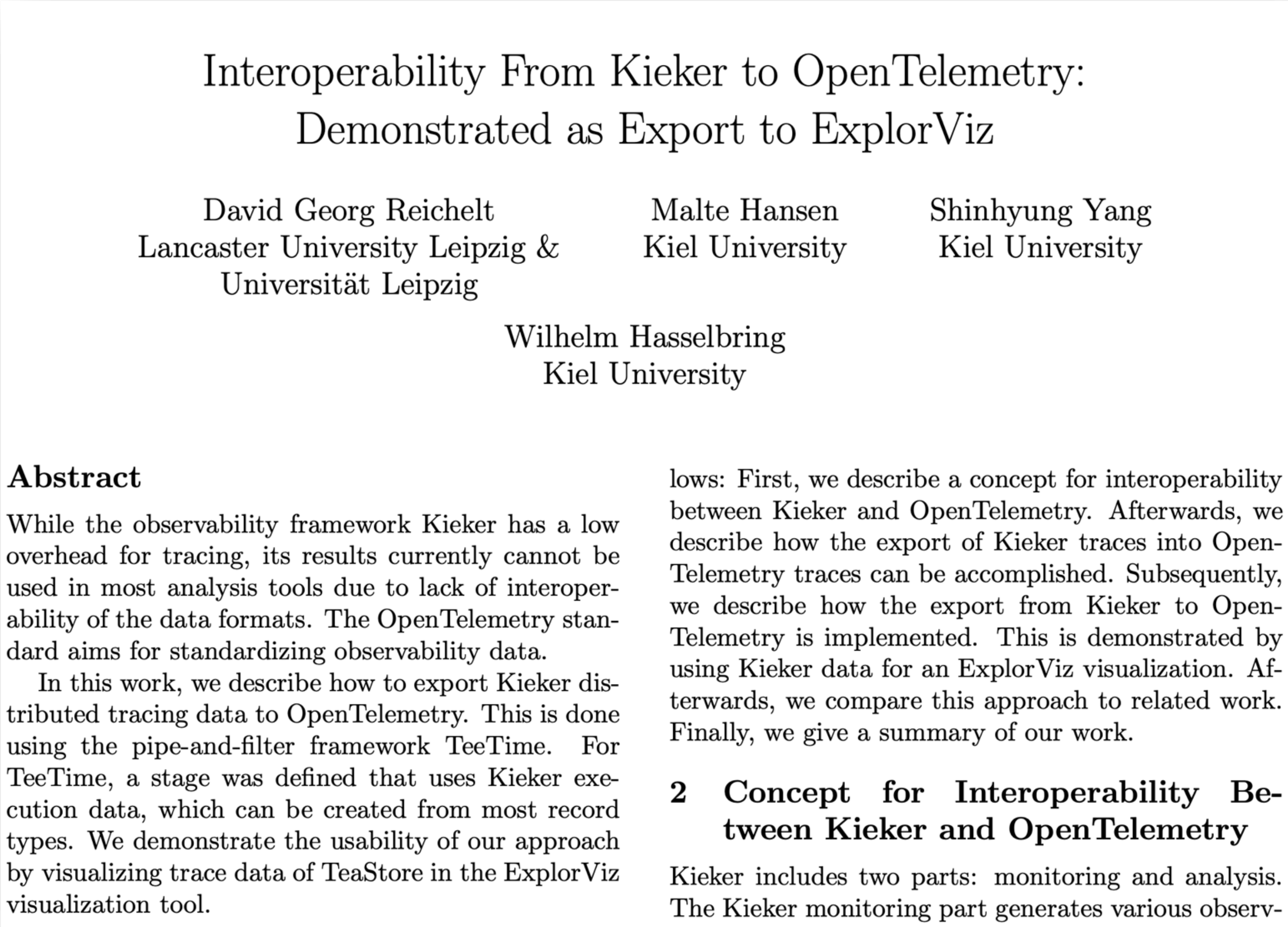


Figure 1: Trace Gathering Process

# Kieker and OpenTelemetry



## Interoperability From Kieker to OpenTelemetry: Demonstrated as Export to ExplorViz

David Georg Reichelt
Lancaster University Leipzig &
Universität Leipzig

Malte Hansen
Kiel University

Shinhyung Yang
Kiel University

Wilhelm Hasselbring
Kiel University

### Abstract

While the observability framework Kieker has a low overhead for tracing, its results currently cannot be used in most analysis tools due to lack of interoperability of the data formats. The OpenTelemetry standard aims for standardizing observability data.

In this work, we describe how to export Kieker distributed tracing data to OpenTelemetry. This is done using the pipe-and-filter framework TeeTime. For TeeTime, a stage was defined that uses Kieker execution data, which can be created from most record types. We demonstrate the usability of our approach by visualizing trace data of TeaStore in the ExplorViz visualization tool.

lows: First, we describe a concept for interoperability between Kieker and OpenTelemetry. Afterwards, we describe how the export of Kieker traces into OpenTelemetry traces can be accomplished. Subsequently, we describe how the export from Kieker to OpenTelemetry is implemented. This is demonstrated by using Kieker data for an ExplorViz visualization. Afterwards, we compare this approach to related work. Finally, we give a summary of our work.

### 2 Concept for Interoperability Between Kieker and OpenTelemetry

Kieker includes two parts: monitoring and analysis. The Kieker monitoring part generates various observ-

[6]

# Demonstration with openHAB Zigbee Binding

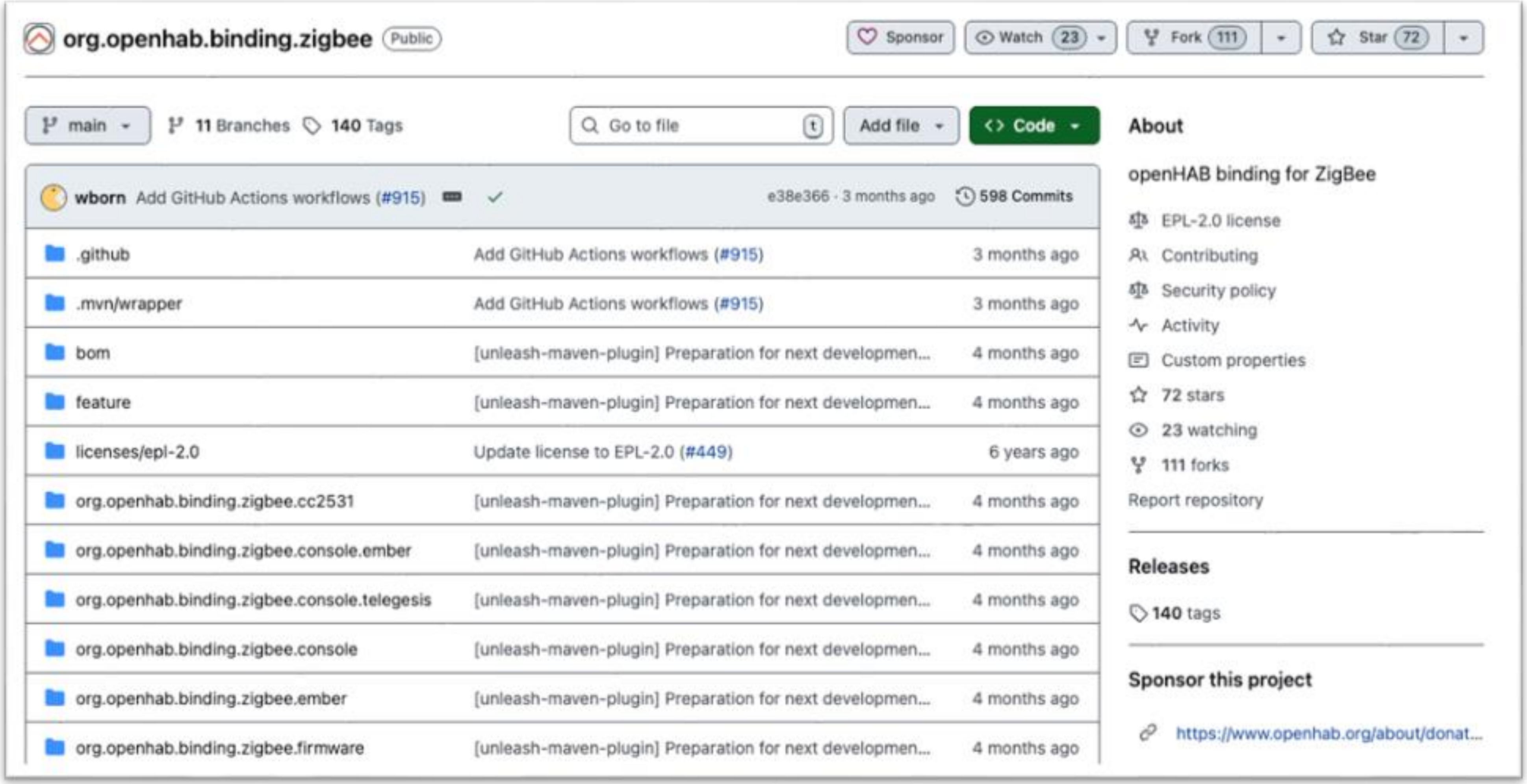
# openHAB binding for ZigBee

- openHAB (open Home Automation Bus)

openHAB
empowering the smart home
a vendor and technology agnostic open source automation software for your home

- ZigBee is a wireless protocol commonly used in smart homes

# openHAB binding for ZigBee



org.openhab.binding.zigbee Public

11 Branches 140 Tags

wborn Add GitHub Actions workflows (#915) ✓ e38e366 · 3 months ago 598 Commits

| | | |
|---|---|---|
| .github | Add GitHub Actions workflows (#915) | 3 months ago |
| .mvn/wrapper | Add GitHub Actions workflows (#915) | 3 months ago |
| bom | [unleash-maven-plugin] Preparation for next developmen... | 4 months ago |
| feature | [unleash-maven-plugin] Preparation for next developmen... | 4 months ago |
| licenses/epl-2.0 | Update license to EPL-2.0 (#449) | 6 years ago |
| org.openhab.binding.zigbee.cc2531 | [unleash-maven-plugin] Preparation for next developmen... | 4 months ago |
| org.openhab.binding.zigbee.console.ember | [unleash-maven-plugin] Preparation for next developmen... | 4 months ago |
| org.openhab.binding.zigbee.console.telegesis | [unleash-maven-plugin] Preparation for next developmen... | 4 months ago |
| org.openhab.binding.zigbee.console | [unleash-maven-plugin] Preparation for next developmen... | 4 months ago |
| org.openhab.binding.zigbee.ember | [unleash-maven-plugin] Preparation for next developmen... | 4 months ago |
| org.openhab.binding.zigbee.firmware | [unleash-maven-plugin] Preparation for next developmen... | 4 months ago |

**About**

openHAB binding for ZigBee

- EPL-2.0 license
- Contributing
- Security policy
- Activity
- Custom properties
- 72 stars
- 23 watching
- 111 forks

Report repository

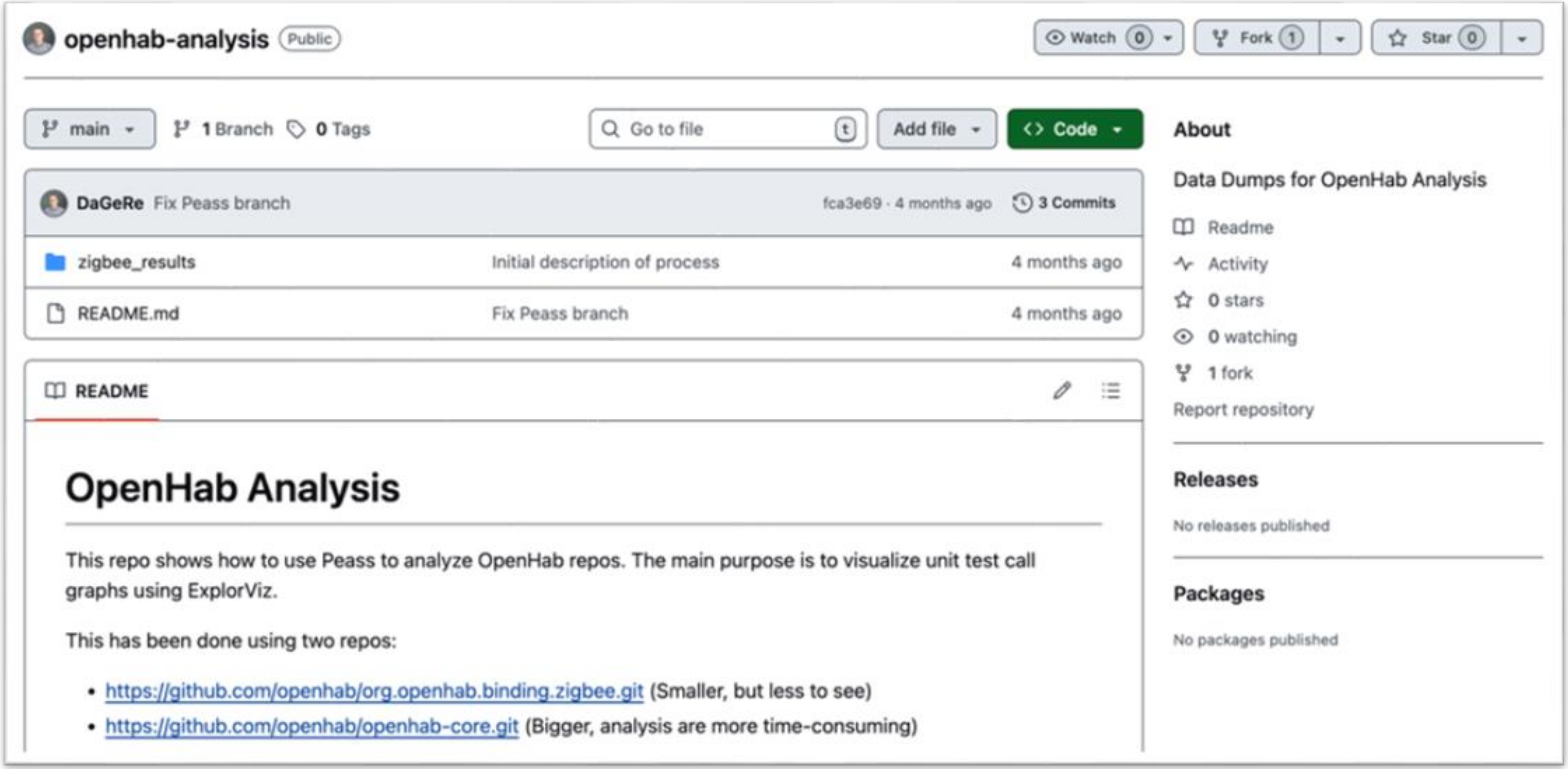
**Releases**

140 tags

**Sponsor this project**

https://www.openhab.org/about/donat...

https://github.com/openhab/org.openhab.binding.zigbee



| | | |
|---|---|---|
| DaGeRe Fix Peass branch | | fca3e69 · 4 months ago · 3 Commits |
| zigbee_results | Initial description of process | 4 months ago |
| README.md | Fix Peass branch | 4 months ago |

# OpenHab Analysis

This repo shows how to use Peass to analyze OpenHab repos. The main purpose is to visualize unit test call graphs using ExplorViz.

This has been done using two repos:

- https://github.com/openhab/org.openhab.binding.zigbee.git (Smaller, but less to see)
- https://github.com/openhab/openhab-core.git (Bigger, analysis are more time-consuming)

**About**

Data Dumps for OpenHab Analysis

Readme

Activity

0 stars

0 watching

1 fork

Report repository

**Releases**

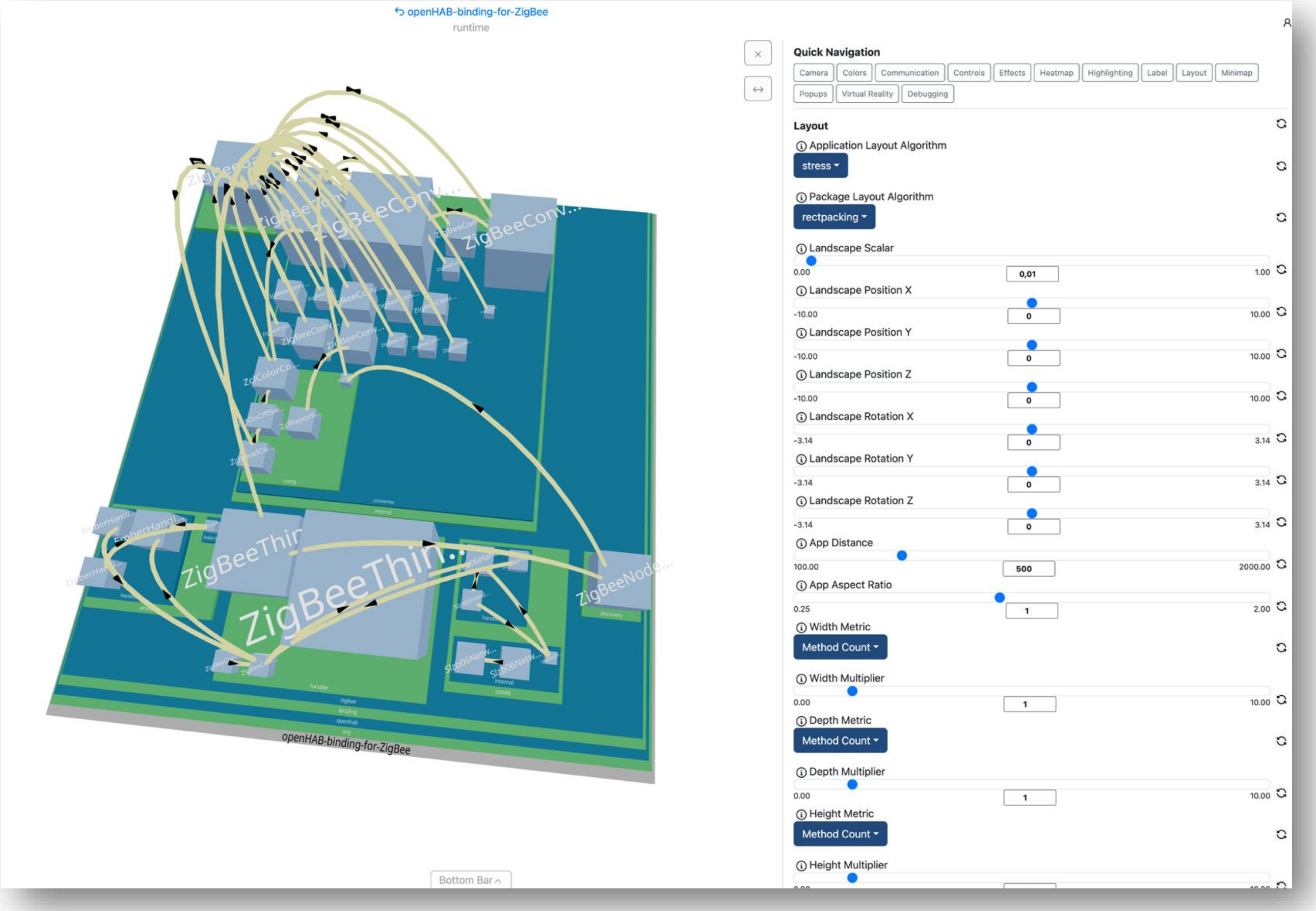
No releases published

**Packages**

No packages published

https://github.com/DaGeRe/openhab-analysis

# Visualization: Dynamic Data Only



Layout

Application Layout Algorithm: stress

Package Layout Algorithm: rectpacking

Landscape Scalar: 0.00 – 1.00: 0,01

Landscape Position X: -10.00 – 10.00: 0

Landscape Position Y: -10.00 – 10.00: 0

Landscape Position Z: -10.00 – 10.00: 0

Landscape Rotation X: -3.14 – 3.14: 0

Landscape Rotation Y: -3.14 – 3.14: 0

Landscape Rotation Z: -3.14 – 3.14: 0

App Distance: 100.00 – 2000.00: 500

App Aspect Ratio: 0.25 – 2.00: 1

Width Metric: Method Count

Width Multiplier: 0.00 – 10.00: 1

Depth Metric: Method Count

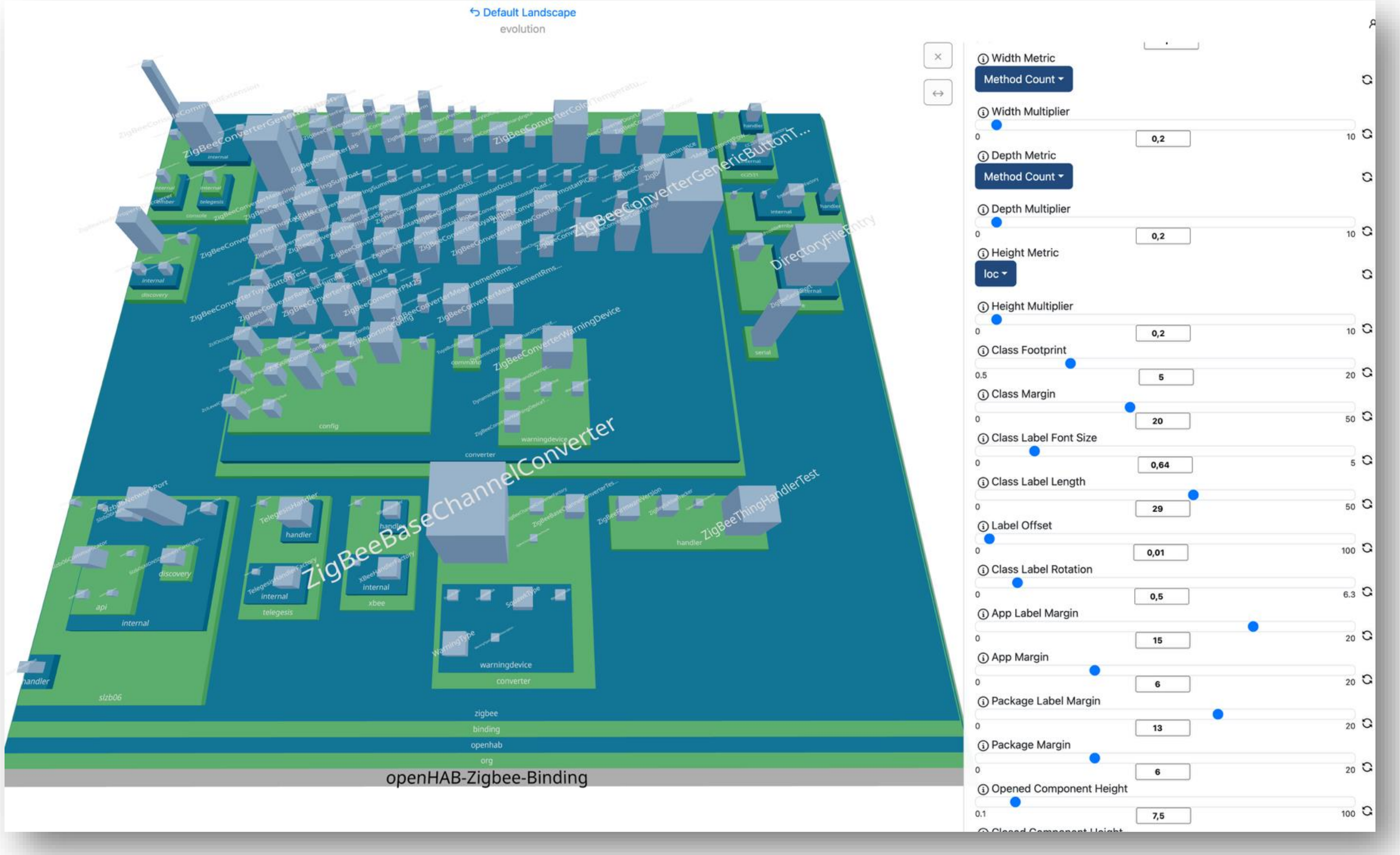
Depth Multiplier: 0.00 – 10.00: 1

Height Metric: Method Count

Height Multiplier

Data for commit: 2f0a43

# Visulization: Static Data Only



Width Metric
Method Count

Width Multiplier
0 0,2 10

Depth Metric
Method Count

Depth Multiplier
0 0,2 10

Height Metric
loc

Height Multiplier
0 0,2 10

Class Footprint
0.5 5 20

Class Margin
0 20 50

Class Label Font Size
0 0,64 5

Class Label Length
0 29 50

Label Offset
0 0,01 100

Class Label Rotation
0 0,5 6.3

App Label Margin
0 15 20

App Margin
0 6 20

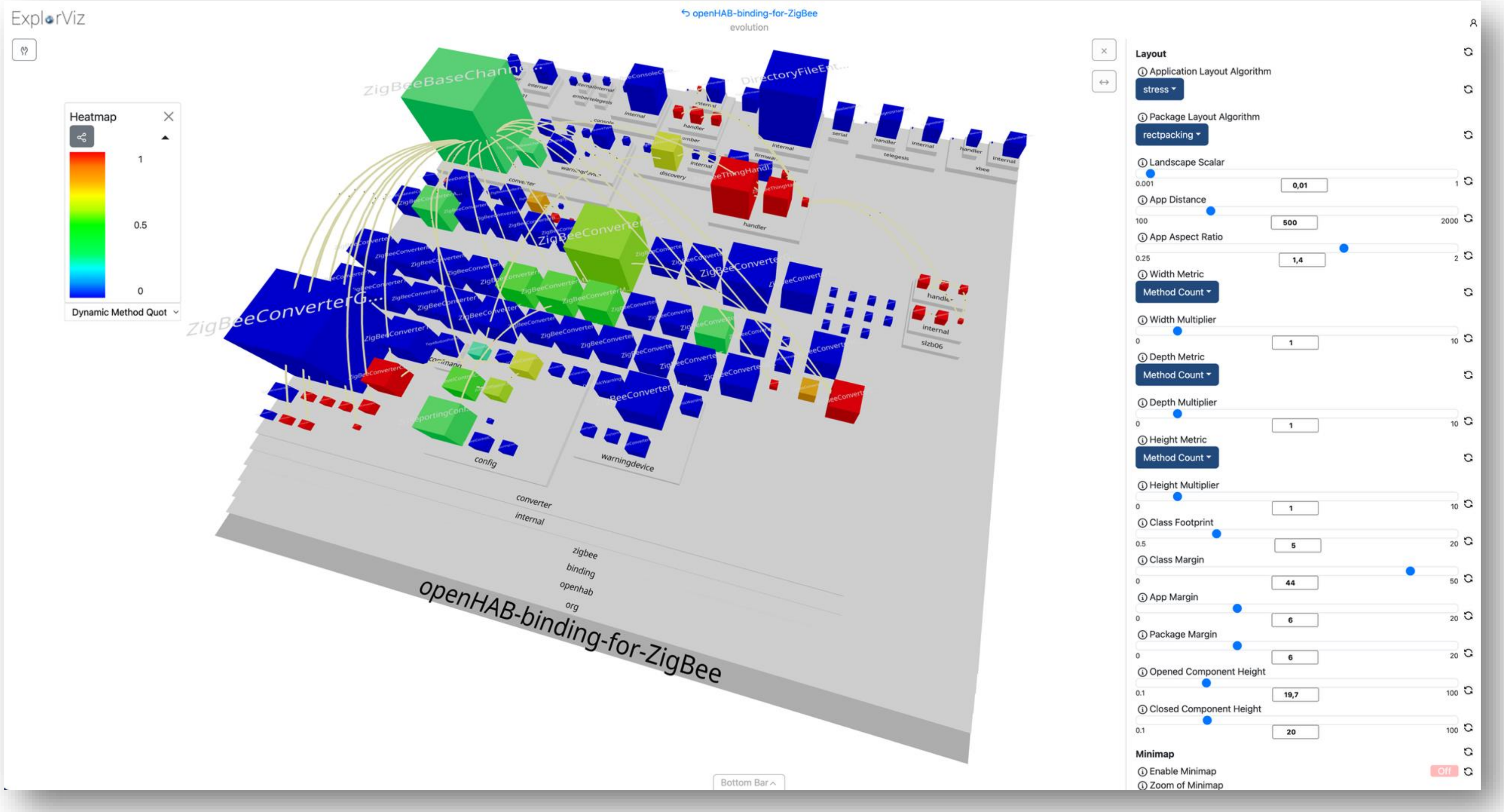
Package Label Margin
0 13 20

Package Margin
0 6 20

Opened Component Height
0.1 7,5 100

Data for commit: 2f0a43

# Test Coverage Visualization



**Layout**

Application Layout Algorithm: stress

Package Layout Algorithm: rectpacking

Landscape Scalar: 0.001 – 1: 0,01

App Distance: 100 – 2000: 500

App Aspect Ratio: 0.25 – 2: 1,4

Width Metric: Method Count

Width Multiplier: 0 – 10: 1

Depth Metric: Method Count

Depth Multiplier: 0 – 10: 1

Height Metric: Method Count

Height Multiplier: 0 – 10: 1

Class Footprint: 0.5 – 20: 5

Class Margin: 0 – 50: 44

App Margin: 0 – 20: 6

Package Margin: 0 – 20: 6

Opened Component Height: 0.1 – 100: 19,7

Closed Component Height: 0.1 – 100: 20

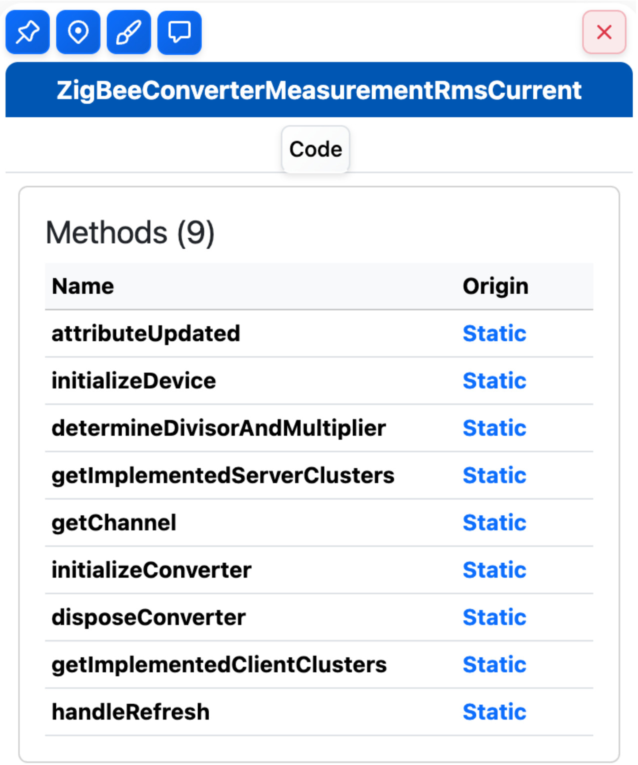
**Minimap**

Enable Minimap: Off

Zoom of Minimap

Data for commit: 2f0a43

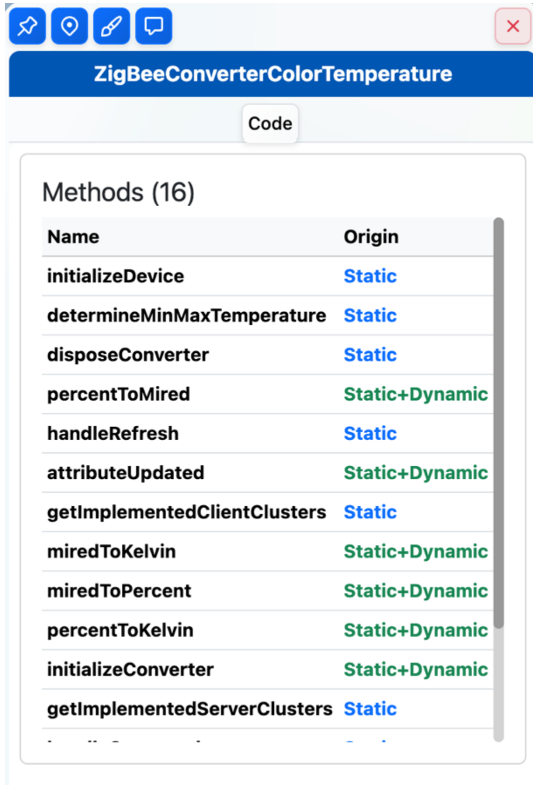
# Class Information in Popovers



ZigBeeConverterMeasurementRmsCurrent

Methods (9)

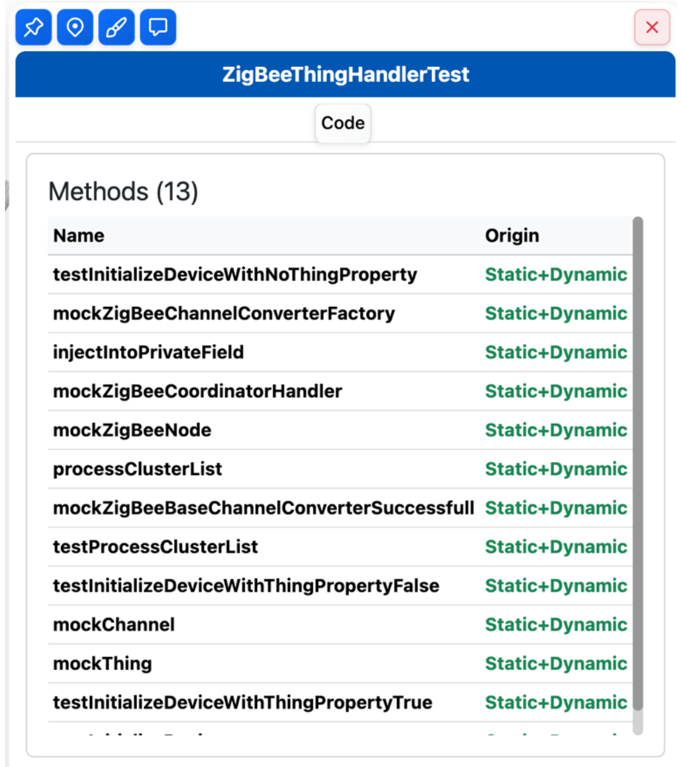
| Name | Origin |
|---|---|
| attributeUpdated | Static |
| initializeDevice | Static |
| determineDivisorAndMultiplier | Static |
| getImplementedServerClusters | Static |
| getChannel | Static |
| initializeConverter | Static |
| disposeConverter | Static |
| getImplementedClientClusters | Static |
| handleRefresh | Static |



ZigBeeConverterColorTemperature

Methods (16)

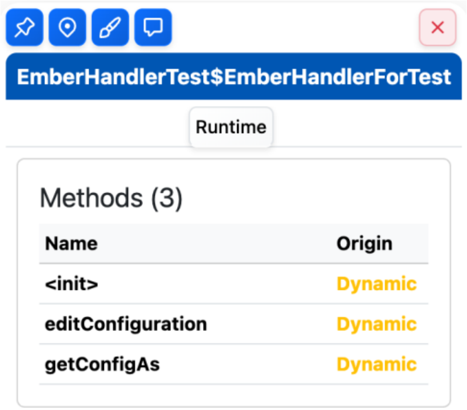
| Name | Origin |
|---|---|
| initializeDevice | Static |
| determineMinMaxTemperature | Static |
| disposeConverter | Static |
| percentToMired | Static+Dynamic |
| handleRefresh | Static |
| attributeUpdated | Static+Dynamic |
| getImplementedClientClusters | Static |
| miredToKelvin | Static+Dynamic |
| miredToPercent | Static+Dynamic |
| percentToKelvin | Static+Dynamic |
| initializeConverter | Static+Dynamic |
| getImplementedServerClusters | Static |



ZigBeeThingHandlerTest

Methods (13)

| Name | Origin |
|---|---|
| testInitializeDeviceWithNoThingProperty | Static+Dynamic |
| mockZigBeeChannelConverterFactory | Static+Dynamic |
| injectIntoPrivateField | Static+Dynamic |
| mockZigBeeCoordinatorHandler | Static+Dynamic |
| mockZigBeeNode | Static+Dynamic |
| processClusterList | Static+Dynamic |
| mockZigBeeBaseChannelConverterSuccessfull | Static+Dynamic |
| testProcessClusterList | Static+Dynamic |
| testInitializeDeviceWithThingPropertyFalse | Static+Dynamic |
| mockChannel | Static+Dynamic |
| mockThing | Static+Dynamic |
| testInitializeDeviceWithThingPropertyTrue | Static+Dynamic |



EmberHandlerTest$EmberHandlerForTest

Methods (3)

| Name | Origin |
|---|---|
| <init> | Dynamic |
| editConfiguration | Dynamic |
| getConfigAs | Dynamic |

=> Did not catch inner class with static analysis

# Summary & Outlook

## Summary

- Combined Kieker, Peass, and ExplorViz to visualize unit test traces and test coverage
- Result: 3D visualization that makes test results explorable in the context of software architecture and evolution

## Outlook

- Focus on the performance analysis and visualization of executed unit tests

# References

- [1]L. Merino, M. Ghafari, C. Anslow, and O. Nierstrasz, “A systematic literature review of software visualization evaluation,” J Syst Software, vol. 144, pp. 165–180, 2018, doi: 10.1016/j.jss.2018.06.027.
- [2] K. Dreef, V. K. Palepu, and J. A. Jones. “Global Overviews of Granular Test Coverage with Matrix Visualizations”. In: 2021 Work-ing Conference on Software Visualization (VIS-SOFT) 00 (2021), pp. 44–54. doi: 10 . 1109 /vissoft52517.2021.00014.
- [3] Richard Wettel and Michele Lanza. 2008. CodeCity: 3D visualization of large-scale software. In Companion of the 30th international conference on Software engineering (ICSE Companion '08). Association for Computing Machinery, New York, NY, USA, 921–922. https://doi.org/10.1145/1370175.1370188
- [4] Wilhelm Hasselbring, Alexander Krause, Christian Zirkelbach (2020): ExplorViz: Research on software visualization, comprehension and collaboration. Software Impacts, Volume 6. DOI https://doi.org/10.1016/j.simpa.2020.100034
- [5] D. G. Reichelt, S. Kühne and W. Hasselbring, "PeASS: A Tool for Identifying Performance Changes at Code Level," 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA, 2019, pp. 1146-1149, doi: 10.1109/ASE.2019.00123.
- [6] Reichelt, David Georg; Hansen, Malte; Yang, Shinhyung; Hasselbring, Wilhelm (2025): Interoperability From Kieker to OpenTelemetry: Demonstrated as Export to ExplorViz. Softwaretechnik-Trends Band 45, Heft 1. Gesellschaft für Informatik e.V.. ISSN: 0720-8928