SmartDelta

Automated Quality Assurance and Optimization in Incremental Industrial Software Systems Development

D5.5 - SmartDelta Visualization Dashboard

Submission date of deliverable: Feb 28, 2025

Edited by: Bilge Özdemir (Dakik), Ural Sezer (Dakik), Ömercan Devran (Arçelik), Martin Hess (Software AG), Yuriy Yevstihnyeyev (Vaadin), Zulqarnain Haider (Alstom), Jean Malm (Mälardalen University), Yazmin Andrea Pabon Guerrero (UC3M), Asif Khan (Ontario Tech), Abhishek Shrestha (Fraunhofer), Andreas Dreschinski (Akkodis), Benedikt Dornauer (University of Innsbruck), Johannes Weinzerl (c.c.com, Austria), Mircea-Cristian Racasan (c.c.com, Austria), Hakan Kılınç (NetRD), Akramul Azim (Ontario Tech)

Project start date	Dec 1, 2021
Project duration	36 (+6) months
Project coordinator	Dr. Mehrdad Saadatmand, RISE Research Institutes of Sweden
Project number & call	20023 - ITEA 3 Call 7
Project website	https://itea4.org/project/smartdelta.html & https://smartdelta.org/

Contributing partners	SmartDelta WP5 partners
Version number	V1.0
Work package	WP5
Work package leader	Bilge Özdemir - DAKIK
Dissemination level	Public
Description	This deliverable reports on the development of the SmartDelta dashboard as well as individual dashboards developed by project partners that visualizes software quality characteristics across versions, identifies problematic areas, and provides optimization recommendations for the SmartDelta project.

Executive Summary

The SmartDelta Visualization Dashboard represents a critical component of the SmartDelta project, serving to monitor and optimize software quality across different versions. This deliverable outlines the development of an interactive, data-driven dashboard that integrates findings and metrics from Work Packages 2 through 4, providing stakeholders with a comprehensive view of software evolution. To present different metrics from different use cases on a unified dashboard, the Vaadin charts library was utilized, enabling the visualization of diverse data representations within a single interface.

The primary objective of the SmartDelta Visualization Dashboard is to illustrate the progression of various quality characteristics, such as code complexity, technical debt, performance bottlenecks, and architectural maintainability. By leveraging cutting-edge technologies like the Vaadin platform and partner-contributed visualization tools, the dashboard offers a flexible and customizable solution for software analytics.

The SmartDelta Visualization Dashboard also showcases a collaborative effort, with partners contributing diverse visualizations and analytical perspectives tailored to their specific use cases. This deliverable reports on these individually developed dashboards, which include code quality dashboards, project similarity metrics, resource utilization analysis, anomaly detection, and technical debt tracking. These contributions highlight the dashboard's versatility in addressing various software analysis needs within the SmartDelta ecosystem.

Overall, this deliverable represents a significant milestone in the project, providing a powerful tool for continuous software improvement, data-driven decision-making, and effective collaboration among partners. By combining the Vaadin-based dashboard with partner-specific visualizations, stakeholders gain access to a comprehensive software analytics solution that fosters data-driven insights and informed decision-making throughout the software development lifecycle.

Table of Contents

1. SmartDelta Dashboard Overview

1.1. Introduction

This section provides a comprehensive overview of the SmartDelta-Dashboard solution, detailing its key components, technical architecture, and collaboration strategies. It explains the design and functionality of the Vaadin Dashboard Component, outlines the underlying data access logic, and offers guidance on adding visualizations and collaborating via the repository. In addition, the section covers licensing details and flexible deployment options to ensure that partners can build, customize, and deploy rich, interactive dashboards tailored to their needs.

1.2. Vaadin Dashboard Component

Within the SmartDelta-Dashboard, the Vaadin Dashboard Component is an important element that streamlines the creation of interactive dashboards. The component was developed within the project scope. It offers a versatile, user-friendly interface supporting both static and dynamic layouts. Developers can define dashboards declaratively for fixed interfaces or leverage dynamic, data-bound configurations that empower end users to adjust widget placement and sizing interactively.

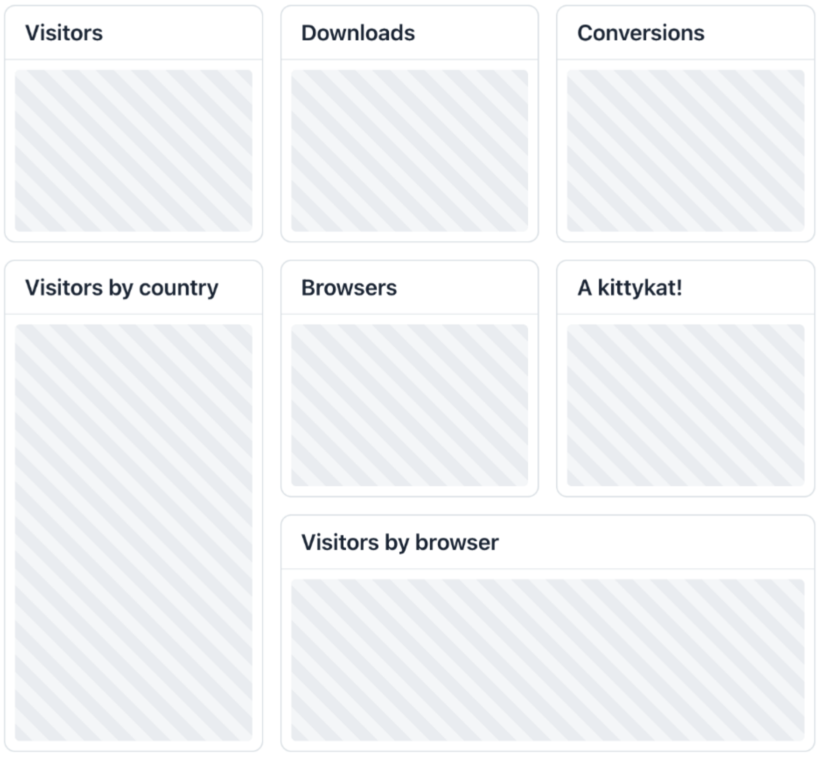


Figure 1. Dashboard component example layout

Key Features:

- **Static & Dynamic Modes:**
    - Supports declarative dashboards for fixed layouts and dynamic dashboards that render widgets based on live data.
    - Dynamic dashboards include an editable mode, enabling users to move, resize, and remove widgets.
- **Responsive Grid Layout:**

- Automatically arranges widgets into adaptive columns and rows.
- As the dashboard's width changes, columns adjust and widget positions reflow accordingly.

- **Intuitive Widget Management:**
  - Provides built-in controls for adjusting column and row spans.
  - In editing mode, widgets can be manipulated via drag & drop, keyboard commands, or accessible move and resize options.
- **Flexible Configuration & Persistence:**
  - Offers comprehensive configuration options including spacing, dense layout preferences, and customizable column/row settings.
  - Supports persistence and reloading of dynamic configurations for personalized, user-specific dashboards.
- **Multi-Platform Integration:**
  - Available for both Java and TypeScript ecosystems with support for React, Lit, and Vaadin Flow, ensuring seamless integration in diverse development environments.
- **Internationalization:**
  - Built-in localization support allows easy adaptation of dashboard text and messages for various languages and regions.

This component delivers a robust and adaptable solution that combines ease of use with powerful customization capabilities. For an in-depth exploration of its technical features and configuration options, please refer to the Vaadin Dashboard Documentation: https://vaadin.com/docs/latest/components/dashboard.

1.3. Technical Architecture and Data Access

The SmartDelta dashboard solution is implemented as a Spring Boot-based web application that leverages a PostgreSQL database, either hosted remotely (e.g., on DigitalOcean) or run locally, to provide a flexible demo and development environment. Each partner is assigned unique credentials tied to a dedicated database schema, ensuring that while collaborators may view others' schemas for context, direct data modifications remain strictly confined to their own areas. Additionally, a special aggregator role is available for secure, read-only access to consolidated views across all partners.

To streamline data management, the project integrates jOOQ, which abstracts complex SQL operations and offers type-safe access to database entities. This integration simplifies routine tasks such as querying records, allowing partners to focus on developing visualizations. Moreover, any changes to the database schema, such as adding or modifying tables, can be easily accommodated by regenerating jOOQ classes using Maven (simply run mvn generate-sources), as detailed in the repository's instructions.

The source-code is available at: https://github.com/vaadin/smartdelta-dashboard/. For access, please contact **yuriy@vaadin.com**.

1.4. Adding Visualizations and Repository Collaboration

Partners can extend the dashboard by adding custom views that display tailored visualizations and metrics. The repository is organized into dedicated packages for each partner (e.g., cc_uibk, software_ag, vaadin, etc.), ensuring a modular structure where UI components (built with Vaadin) and service classes coexist. For example, each package includes a default view along with corresponding service classes, such as CpuNodeService.java in the cc_uibk package, that demonstrate jOOQ usage for efficient data access. This modular structure simplifies an individual partner’s contributions.

Detailed instructions for adding or configuring visualizations are provided within the repository. These guidelines cover tasks from generating jOOQ classes and adjusting Maven configurations to running the application locally.

1.5. Licensing and Deployment Flexibility

Partners have full access to all features of the Vaadin platform, including Vaadin Charts and the aforementioned Vaadin Dashboard components.

Vaadin Charts is a library designed for creating interactive, visually appealing data visualizations in Vaadin applications. It supports a wide range of chart types, including line, bar, pie, scatter, and other chart types, making it suitable for various data presentation needs. With integration into Vaadin applications, it enables dynamic updates, real-time data visualization, and smooth user interactions. Built on top of Highcharts, Vaadin Charts provides extensive customization options, allowing developers to create sophisticated, responsive charts. For further details, please refer to the Vaadin Charts Documentation: https://vaadin.com/docs/latest/components/charts.

Both the Vaadin Charts and Dashboard components are commercial products, with commercial licenses provided to the partners for the duration of the project. This arrangement ensures that partners have access to the advanced features offered by the platform.

In addition to the shared demo setup, partners can deploy local versions of the application. By adjusting database connection settings, such as switching from a remote DigitalOcean database to a local instance, and, if needed, removing views from other collaborators, partners can work with their own real data. This enables the application to be used in a local production environment for visualizing specific metrics without sharing them with other partners. Such flexibility supports both collaborative demonstrations and isolated production deployments.

2. Visualizations of SmartDelta Dashboard

This section presents a collection of visualization examples contributed by our partners, serving as demonstration prototypes that showcase the dashboard’s analytical and interactive capabilities. Each visualization is developed using simulated datasets that mirror realistic scenarios, thereby highlighting potential analytical perspectives and design flexibility without compromising sensitive information. These examples illustrate how diverse data representations can be achieved using the SmartDelta Dashboard, while visualizations incorporating actual operational data remain securely hosted within each partner’s local production environment to ensure data confidentiality and compliance.

2.1. Software AG

Visualizations Used: Bar and line charts
Illustrates: Code quality trends, Code rework recommended, Commit impact analysis, Code and design repetition & reuse potential
Technology Stack: Vaadin platform, Atlassian Jira, SonarQube
Key Features:
The dashboard shows KPIs relevant to the management grouped into the following categories:

- **Code Quality**: Current size of code base, overall code quality and its trend over time for selected products. Allows to assess the overall complexity and quality of products and their evolution over time.
- **Code Rework Recommendations**: Size of low-quality code per product that might require rework and corresponding trend over time. Allows to estimate the amount of rework needed to improve code quality.
- **Commit Impact Analysis**: Number of code commits detrimentally affecting code quality and their frequency over time. Allows to assess the quality of latest commits and track the quality of the development activities over time.
- **Repetition and Reuse:** Number of solved issues with similar descriptions implemented with similar or different code as well as number of open issues similar to already solved ones. Allows to estimate the amount of repetition done so far and the available potential for reuse.

Dashboard:

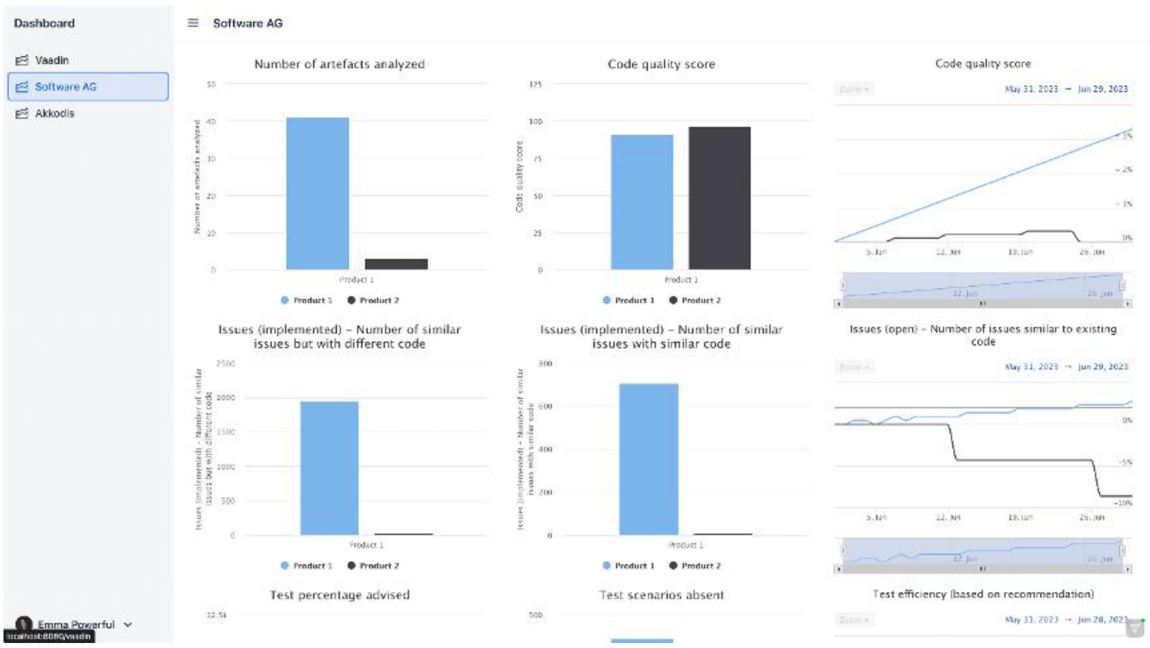

Dashboard
Vaadin
Software AG
Akkodis
Software AG
Number of artefacts analyzed
Code quality score
Code quality score
Product 1
Product 2
Issues (implemented) – Number of similar issues but with different code
Issues (implemented) – Number of similar issues with similar code
Issues (open) – Number of issues similar to existing code
Test percentage advised
Test scenarios absent
Test efficiency (based on recommendation)
May 31, 2023 → Jun 29, 2023
Emma Powerful

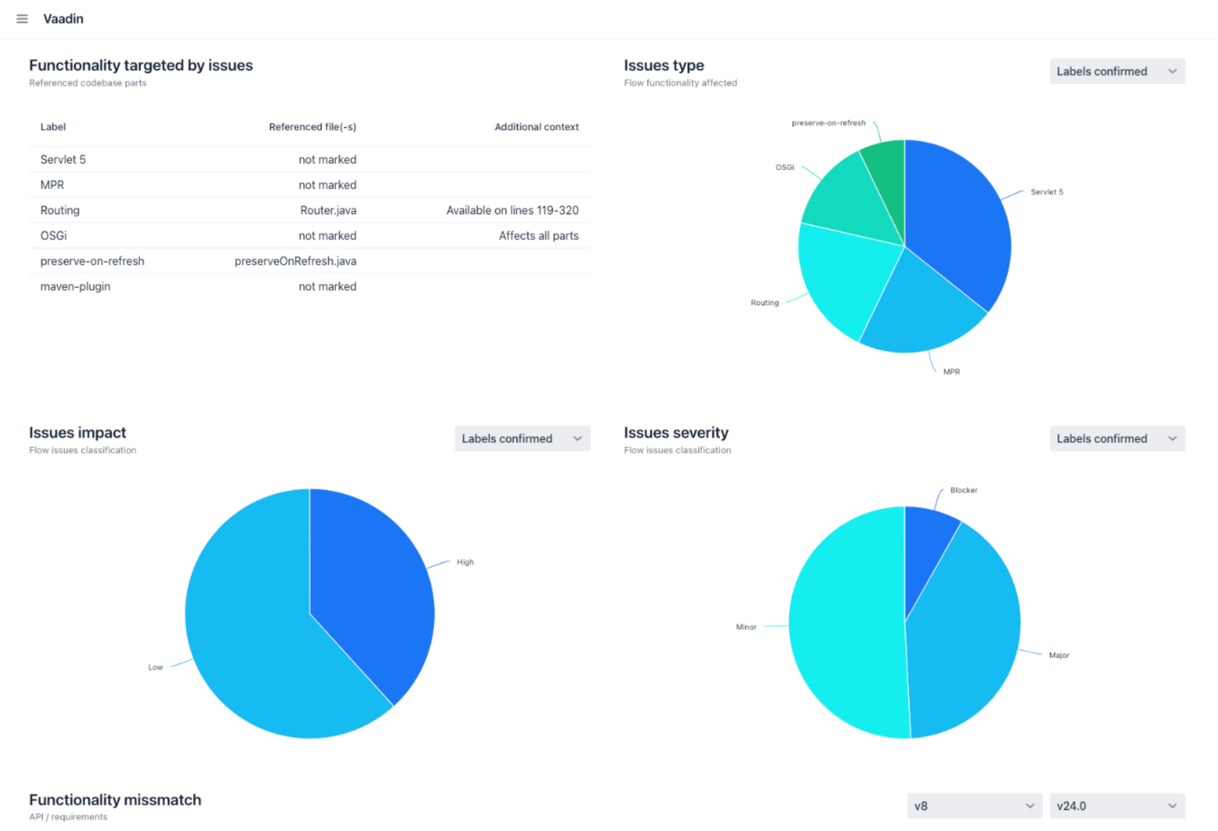
Figure 2. Prototype of the SAG dashboard

2.2. Vaadin

Visualizations Used: Pie charts, bar charts, timelines, spline charts, and grid views.
Illustrates: Issue classifications (by type, impact, and severity), API comparisons, refactor suggestions, and performance trends.
Technology Stack: Vaadin Platform
Key Features:

- **Overview Metrics**: Displays summaries of issue classifications by type, severity, and impact, as well as progress on API migrations and refactoring activities.
- **Interactive Visualizations**: Pie charts, bar graphs, and spline charts enable users to monitor trends, identify bottlenecks, and make informed decisions.
- **Updates Summary**: Highlights changes and areas requiring attention, streamlining prioritization during development cycles.

Dashboard:



Label	Referenced file(-s)	Additional context
Servlet 5	not marked	
MPR	not marked	
Routing	Router.java	Available on lines 119-320
OSGi	not marked	Affects all parts
preserve-on-refresh	preserveOnRefresh.java	
maven-plugin	not marked	

Figure 3. Prototype of the Vaadin dashboard

2.3. cc.com & UIBK

Visualizations Used: Line charts, time-series graph, data distribution plots
Illustrates:

- Evolution of Technical Debt and Code Quality Issues (SoHist Analysis)
- Resource Utilization of Time-Series Data (BLIDS Sensors, Server Cluster)

Technologies:

- Apexcharts for SoHist
- Seaborn for long - time data evaluation
- Grafana for short - time data visualisation
- Vaadin Charts for testing its functionalities

Key Features:

- Track software quality evolution by visualizing e.g., technical debt, code smell density, and test coverage over time
- Analyse large-scale historical and real-time data using time-series graphs and statistical modeling
- Detect anomalies in system behavior through real-time monitoring and rule-based alerting

Visualizations for new Insights:

- SoHist with Apexcharts

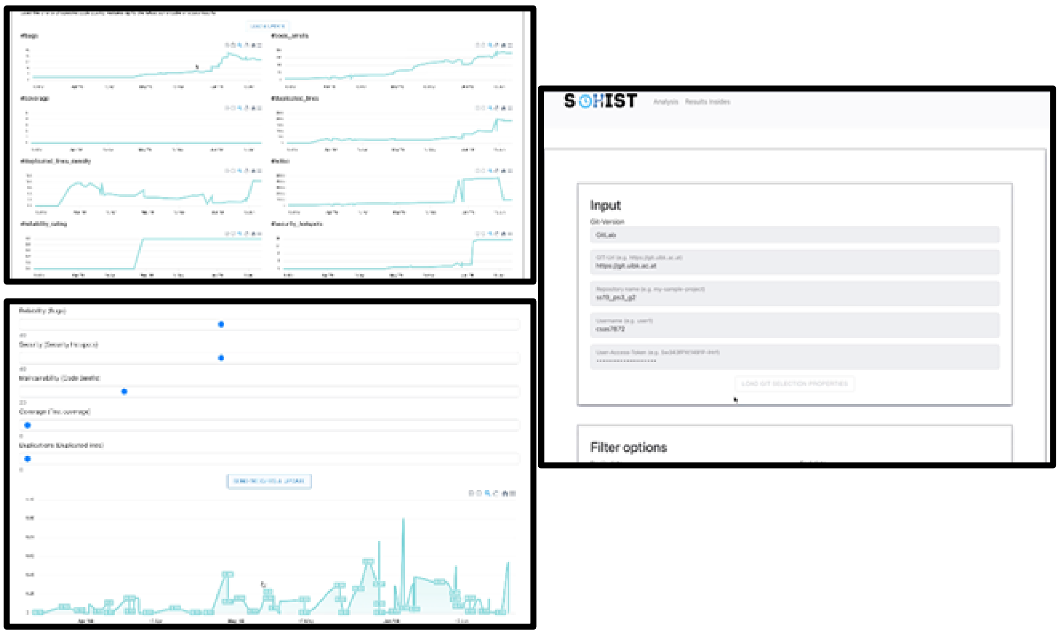


Figure 4. Example Project SoHist (UIBK)

- Seaborn for long - time data evaluation

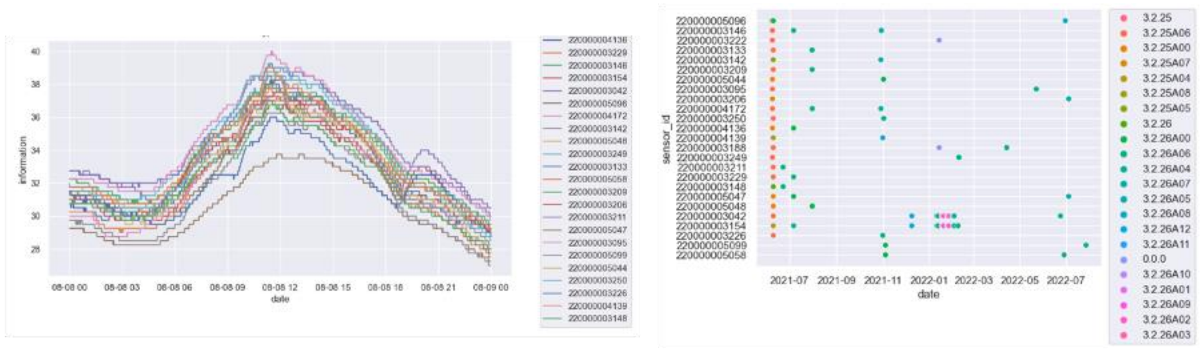
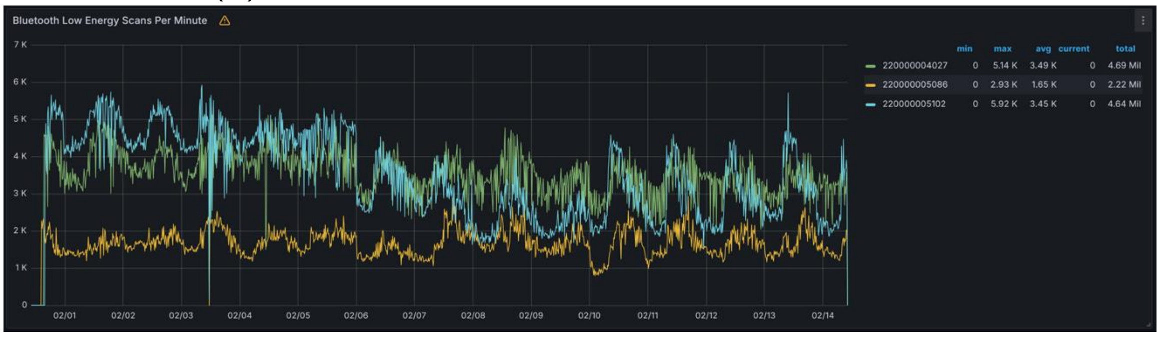


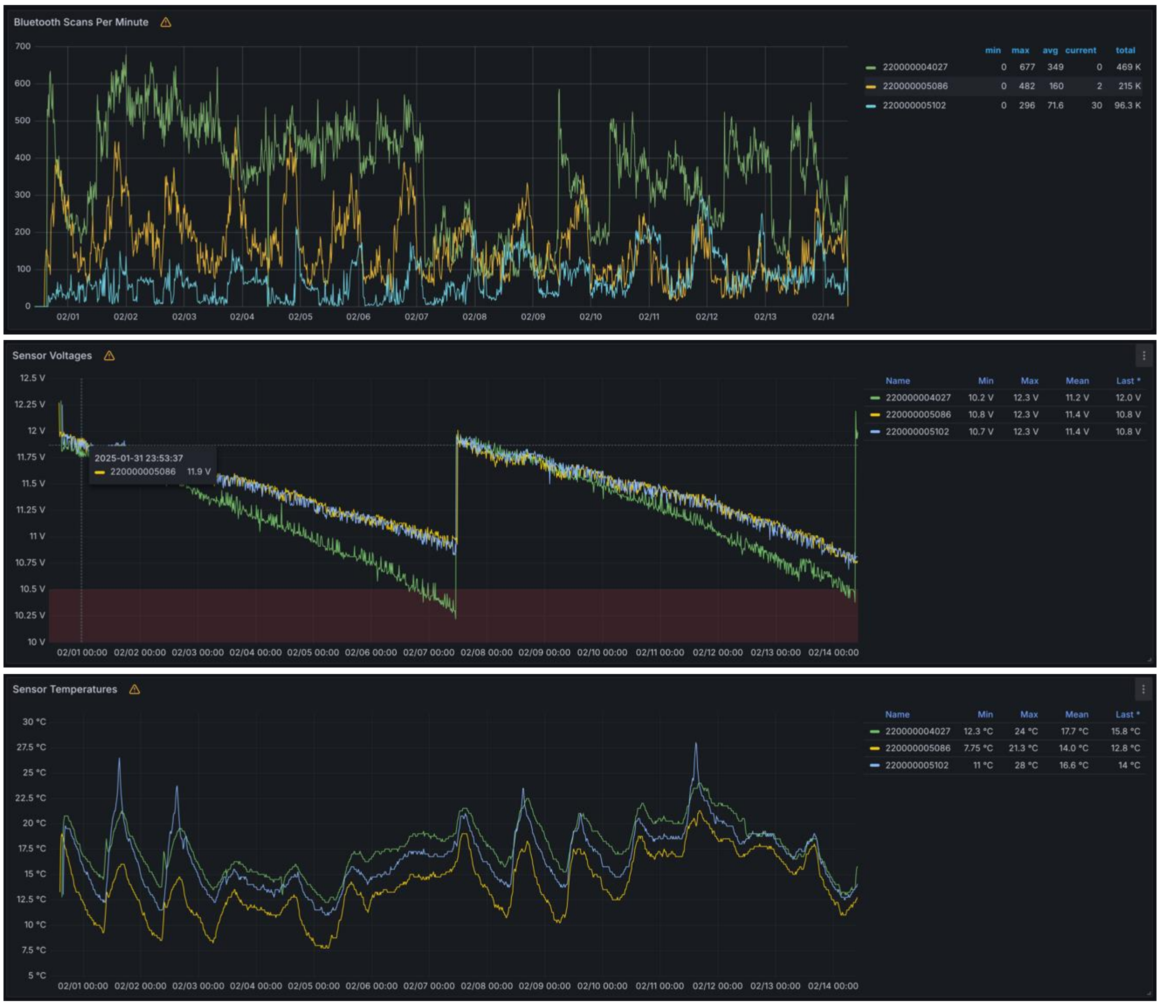
Figure 5. Visualising sensor properties with Python Libraries (left: temperature per sensor, right: sensor version changes over a year)

Dashboard:

- Grafana for short(er) - time data visualization



	min	max	avg	current	total
220000004027	0	5.14 K	3.49 K	0	4.69 Mil
220000005086	0	2.93 K	1.65 K	0	2.22 Mil
220000005102	0	5.92 K	3.45 K	0	4.64 Mil



	min	max	avg	current	total
220000004027	0	677	349	0	469 K
220000005086	0	482	160	2	215 K
220000005102	0	296	71.6	30	96.3 K

Name	Min	Max	Mean	Last *
220000004027	10.2 V	12.3 V	11.2 V	12.0 V
220000005086	10.8 V	12.3 V	11.4 V	10.8 V
220000005102	10.7 V	12.3 V	11.4 V	10.8 V

Name	Min	Max	Mean	Last *
220000004027	12.3 °C	24 °C	17.7 °C	15.8 °C
220000005086	7.75 °C	21.3 °C	14.0 °C	12.8 °C
220000005102	11 °C	28 °C	16.6 °C	14 °C

Figure 6. Visualization for the overview of the sensors

- Testing Vaadin Charts

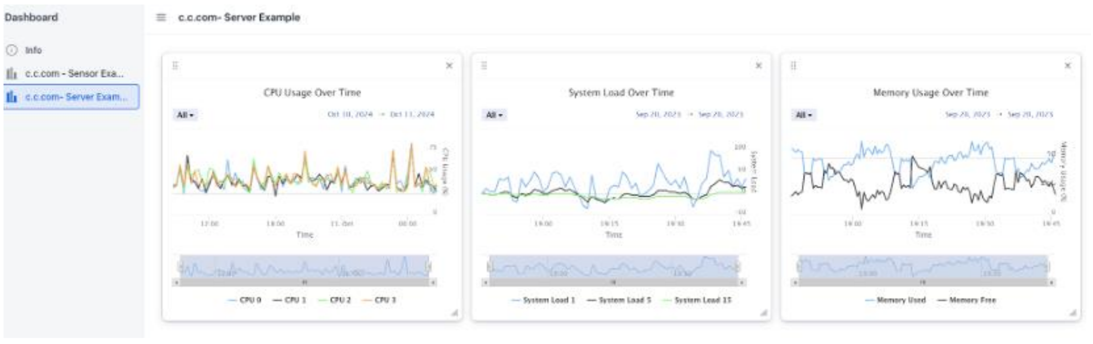


Figure 7. Vaadin charts visualising how individual commits on the main branch affects the code quality

2.4. Izertis

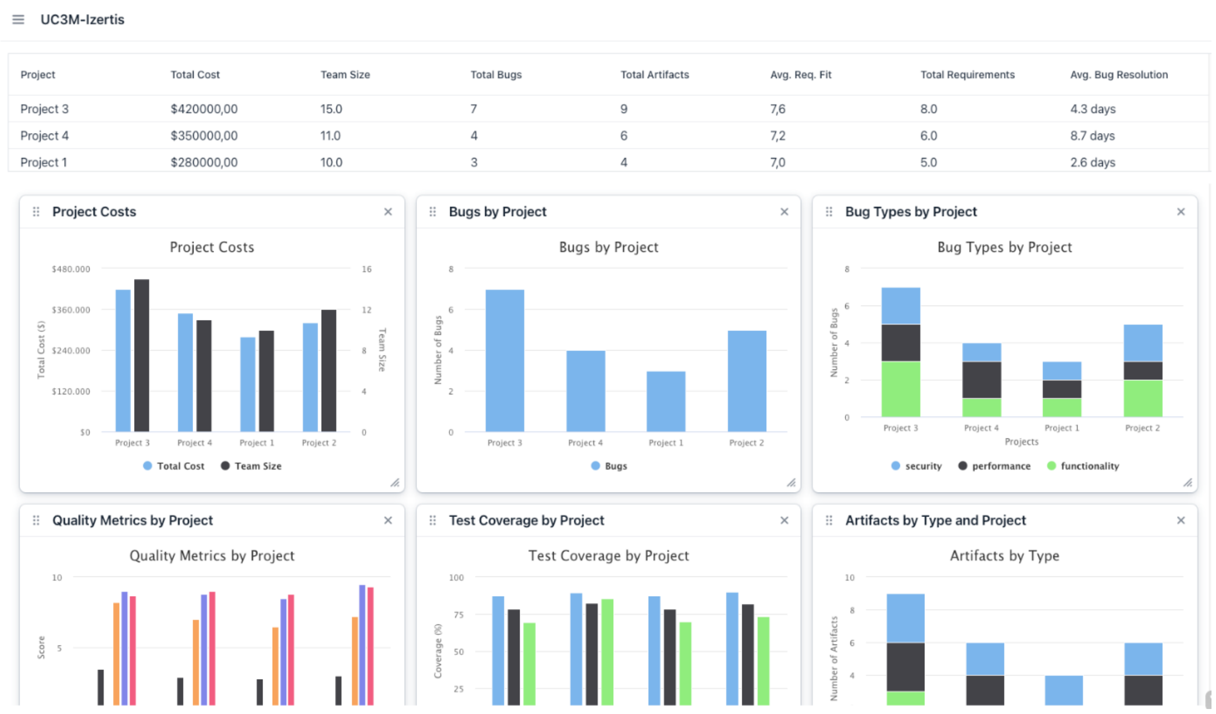
Visualization Used: Bar and line charts
Illustrates: Project similarity metrics, quality metrics, artifact reusability, test effectiveness
Technology Stack: SONATA, Vaadin platform
Key Features:

- Project Similarity Analysis
    - Displays matching requirements between current and historical projects

- Shows artifact distribution across similar projects by type (project management, code, documentation)
- Presents defect patterns and classifications from comparable projects

- Quality Metrics Overview
  - Visualizes defect distributions by category (functionality, security, performance)
  - Tracks project costs and resource allocation across similar implementations
  - Enables custom metric definition and monitoring for specific project needs
- Interactive Reporting
  - Provides detailed artifact analysis with filtering capabilities
  - Presents bug reports and correlation with project characteristics
  - Offers customizable views for different stakeholder needs

Dashboard:


UC3M-Izertis

Project	Total Cost	Team Size	Total Bugs	Total Artifacts	Avg. Req. Fit	Total Requirements	Avg. Bug Resolution
Project 3	$420000,00	15.0	7	9	7,6	8.0	4.3 days
Project 4	$350000,00	11.0	4	6	7,2	6.0	8.7 days
Project 1	$280000,00	10.0	3	4	7,0	5.0	2.6 days

Figure 8. General view of Izertis Dashboard

3. Individual Visualizations of SmartDelta Project

3.1. Alstom

Visualizations Used: Code Quality Dashboards; bar charts, tables and graphical renderings of models.

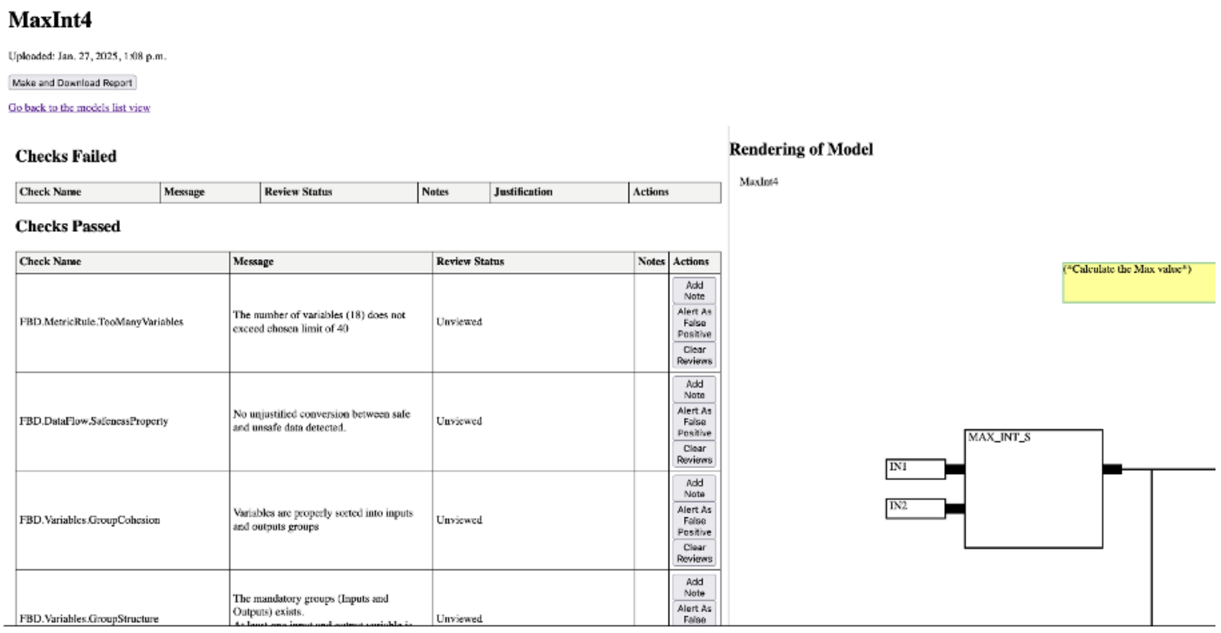
Illustrates: Code guideline violations. Nr of models passing and failing checks, software deltas

Technology Stack: DRACONIS (python, Django)

Key Features:

- Overview of the quality of a given set of block models.
- Graphical and semantic deltas between model variants.
- Fast feedback to stakeholders through edit-time check reports (development) and downloadable excel documents (review process).

Dashboard:



MaxInt4

Uploaded: Jan. 27, 2025, 1:08 p.m.

Make and Download Report

Go back to the models list view

Checks Failed

Check Name	Message	Review Status	Notes	Justification	Actions

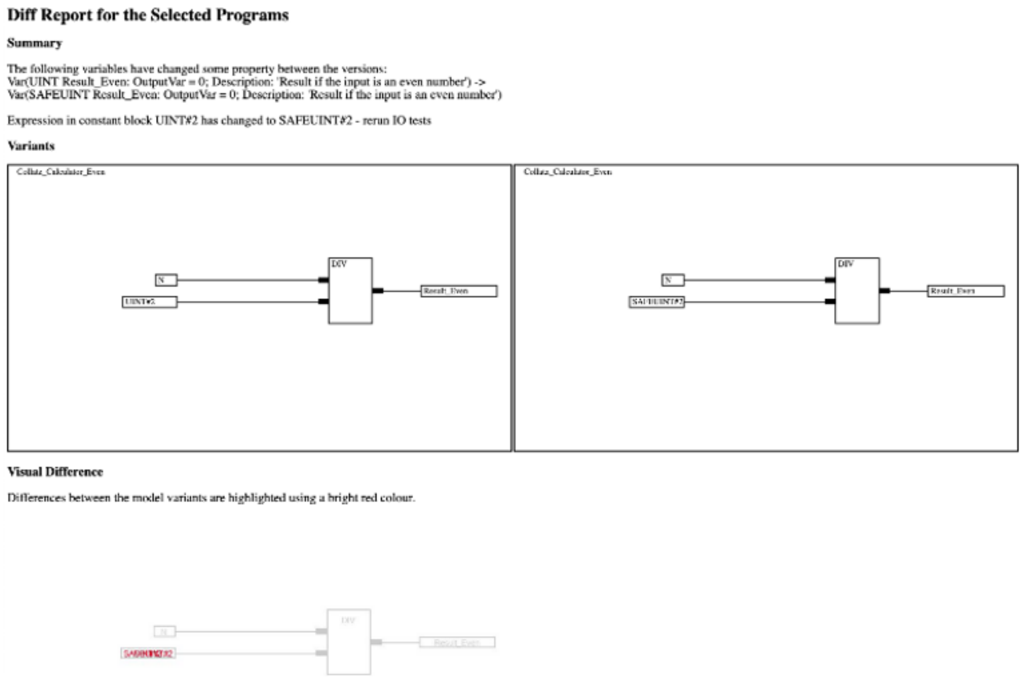
Checks Passed

Check Name	Message	Review Status	Notes	Actions
FBD.MetricRule.TooManyVariables	The number of variables (18) does not exceed chosen limit of 40	Unviewed		Add Note / Alert As False Positive / Clear Reviews
FBD.DataFlow.SafenessProperty	No unjustified conversion between safe and unsafe data detected.	Unviewed		Add Note / Alert As False Positive / Clear Reviews
FBD.Variables.GroupCohesion	Variables are properly sorted into inputs and outputs groups	Unviewed		Add Note / Alert As False Positive / Clear Reviews
FBD.Variables.GroupStructure	The mandatory groups (Inputs and Outputs) exists.	Unviewed		Add Note / Alert As False

Rendering of Model

MaxInt4

Figure 9. Report view for a block model



Diff Report for the Selected Programs

Summary

The following variables have changed some property between the versions:
Var(UINT Result_Even: OutputVar = 0; Description: 'Result if the input is an even number') ->
Var(SAFEUINT Result_Even: OutputVar = 0; Description: 'Result if the input is an even number')

Expression in constant block UINT#2 has changed to SAFEUINT#2 - rerun IO tests

Variants

Visual Difference

Differences between the model variants are highlighted using a bright red colour.

Figure 10. Delta visualization of two variants

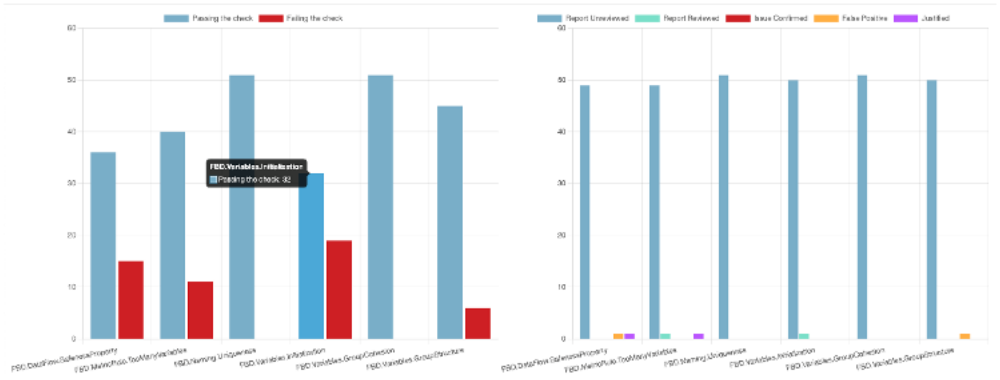


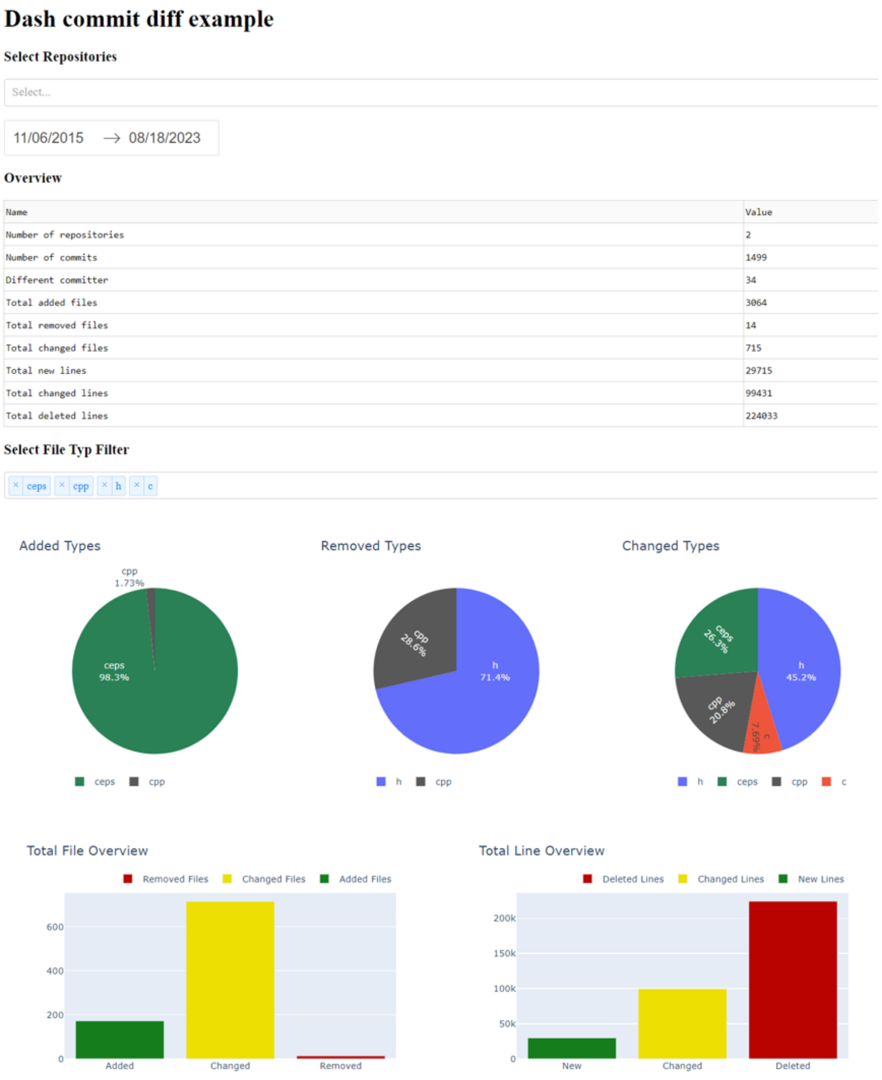
Figure 11. Overview of the ongoing review status

3.2. Akkodis

Visualizations Used: Pie charts; bar charts, Clustering map, UML state diagrams, custom visualization types (Execution Flow State Diagram, Log Similarity Matrix, Event Flow Diagram)
Illustrates: Repository insights, software delta & evolution
Technology Stack: PlantUML, Streamlit, dash/plotly
Key Features:

- Compare git commits
- Show repository insights and clustering of artefacts
- Identify similarities or deltas among models
- Generate models from requirements
- The Architecture Analysis and Visualization Dashboard (ref. Section 2.3) computes and visualizes various architectural views.

Dashboard:



Dash commit diff example

Select Repositories

Select...

11/06/2015 → 08/18/2023

Overview

Name	Value
Number of repositories	2
Number of commits	1499
Different committer	34
Total added files	3064
Total removed files	14
Total changed files	715
Total new lines	29715
Total changed lines	99431
Total deleted lines	224033

Select File Typ Filter

× ceps × cpp × h × c

Figure 12. Prototype of dashboard showing repository insights with delta view

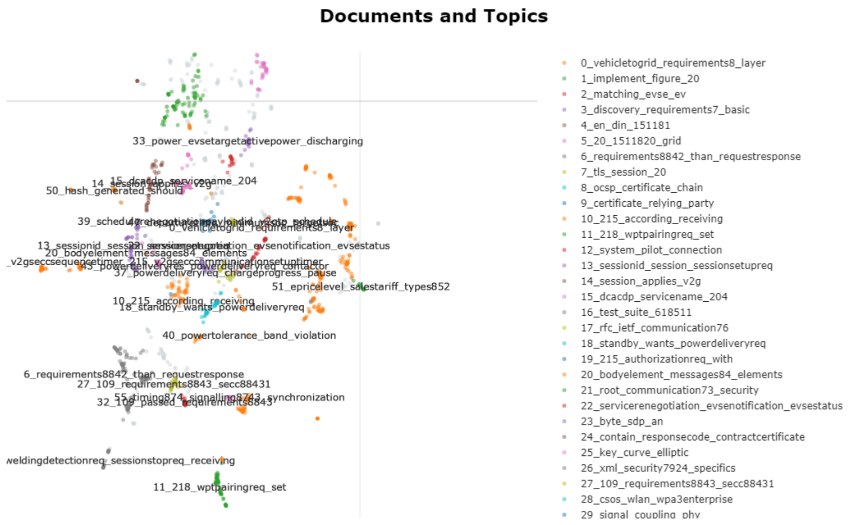


Figure 13. Prototype showing clustering of repository artefacts using topic modelling

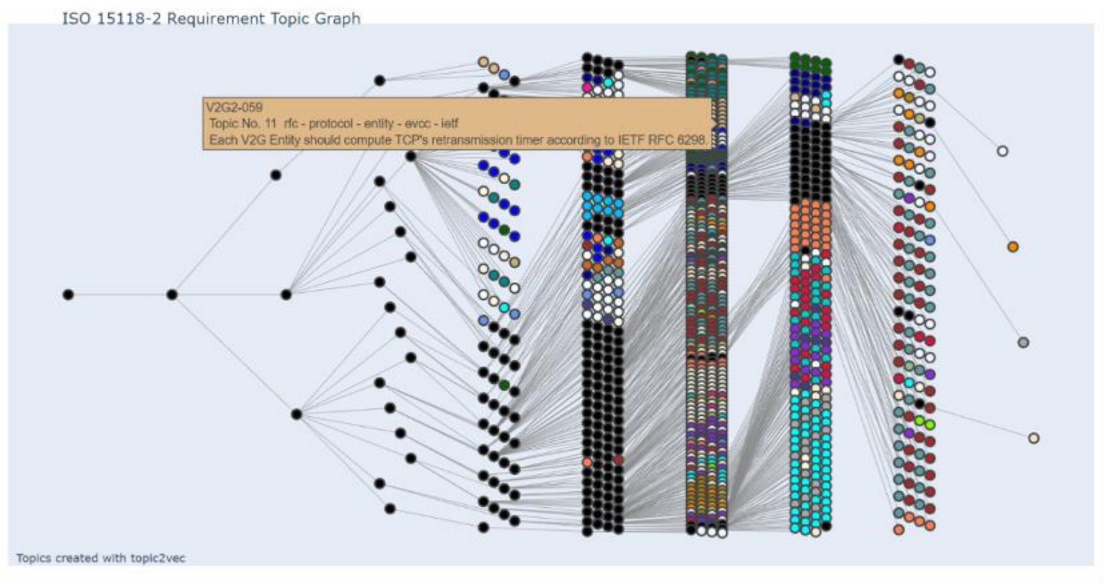

ISO 15118-2 Requirement Topic Graph
V2G2-059
Topic No. 11 rfc - protocol - entity - evcc - ietf
Each V2G Entity should compute TCP's retransmission timer according to IETF RFC 6298.
Topics created with topic2vec

Figure 14. Prototype using custom visualization showing semantic structure of a document (ISO 15118-2)

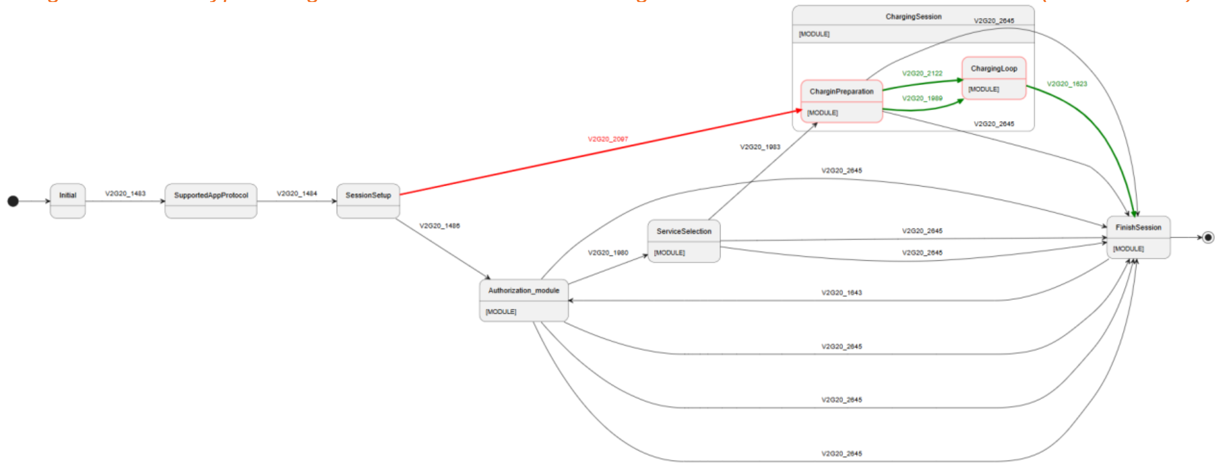


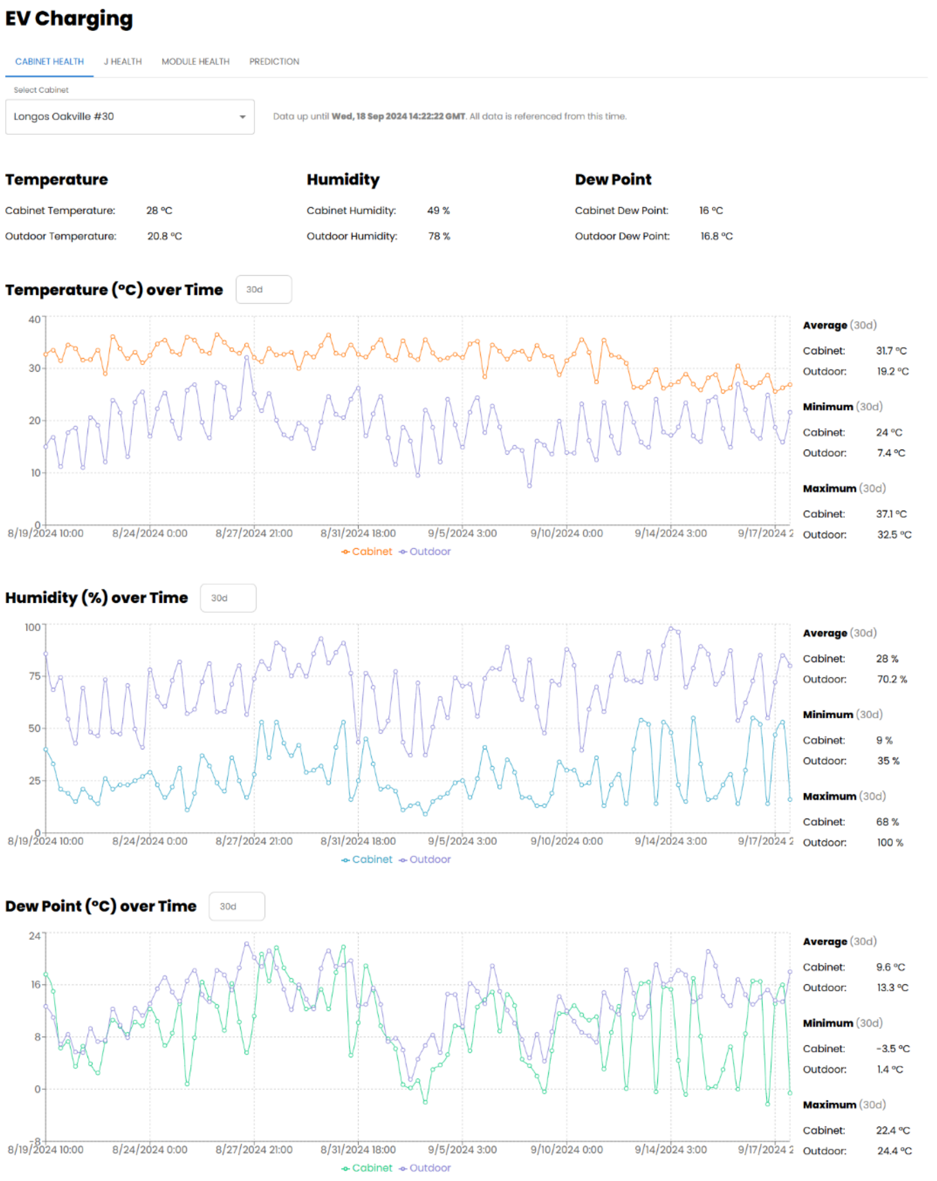
Figure 15. UML state machine visualization with delta highlighting

3.3. eCamion

Visualizations Used: line graph, histogram, box plot diagram
Illustrates: Health and performance of charging stations, anomalies, failure detection
Technology Stack: JavaScript libraries (MaterialUI, Axios, ReCharts, and ChartJS), NodeJS, ReactJS, Python Flask, PostgreSQL database.
Key Features:

- Display charging station usage
- Charging Station cell health and cabinet health, alert to notify abnormal sensor reading
- Prediction of hourly energy delivered by charging station

Dashboard:



EV Charging

CABINET HEALTH | J HEALTH | MODULE HEALTH | PREDICTION

Select Cabinet: Longos Oakville #30

Data up until **Wed, 18 Sep 2024 14:22:22 GMT**. All data is referenced from this time.

Temperature

Cabinet Temperature:	28 °C
Outdoor Temperature:	20.8 °C

Humidity

Cabinet Humidity:	49 %
Outdoor Humidity:	78 %

Dew Point

Cabinet Dew Point:	16 °C
Outdoor Dew Point:	16.8 °C

Temperature (°C) over Time 30d

Average (30d)	
Cabinet:	31.7 °C
Outdoor:	19.2 °C
Minimum (30d)	
Cabinet:	24 °C
Outdoor:	7.4 °C
Maximum (30d)	
Cabinet:	37.1 °C
Outdoor:	32.5 °C

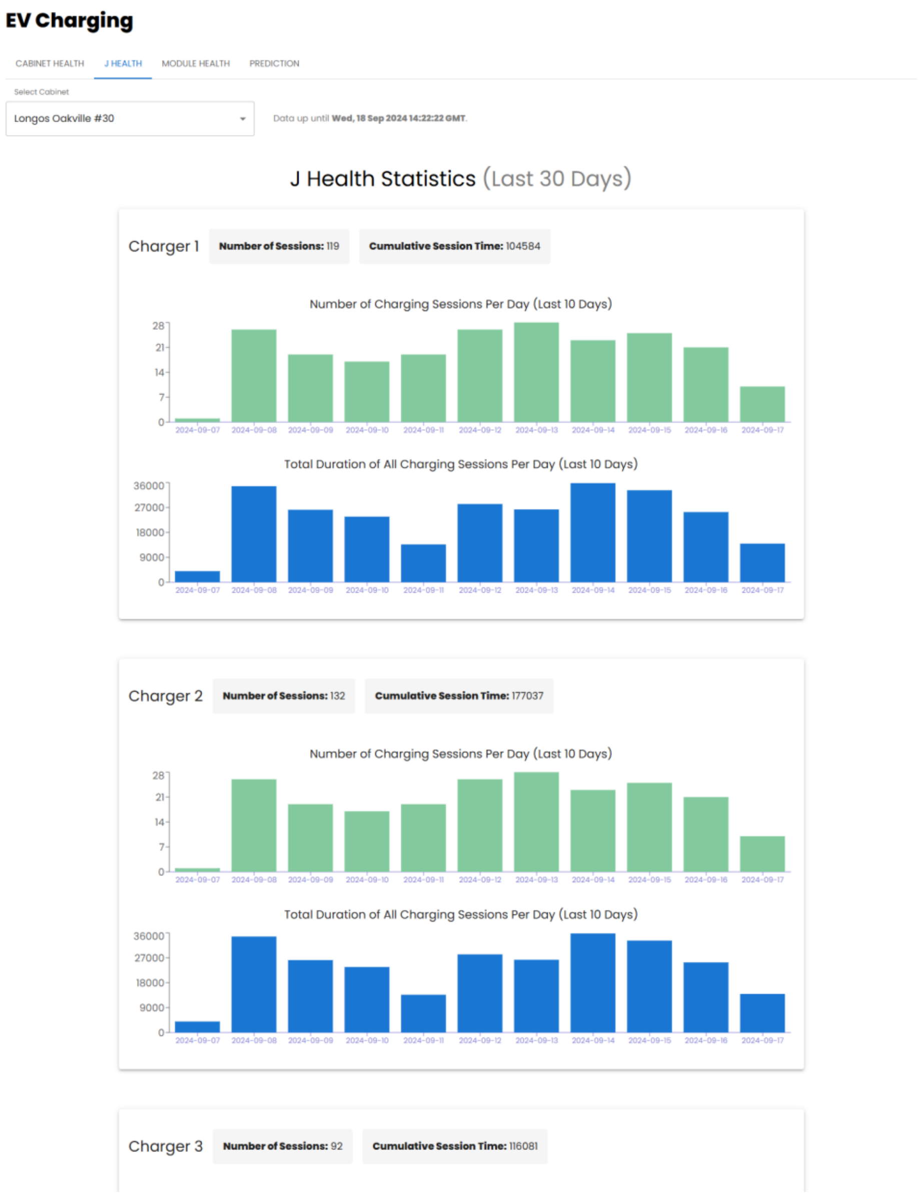
Humidity (%) over Time 30d

Average (30d)	
Cabinet:	28 %
Outdoor:	70.2 %
Minimum (30d)	
Cabinet:	9 %
Outdoor:	35 %
Maximum (30d)	
Cabinet:	68 %
Outdoor:	100 %

Dew Point (°C) over Time 30d

Average (30d)	
Cabinet:	9.6 °C
Outdoor:	13.3 °C
Minimum (30d)	
Cabinet:	-3.5 °C
Outdoor:	1.4 °C
Maximum (30d)	
Cabinet:	22.4 °C
Outdoor:	24.4 °C

Figure 16. Charging station cabinet health overview shown in eCamion Dasboards



EV Charging

Longos Oakville #30

Data up until **Wed, 18 Sep 2024 14:22:22 GMT**.

J Health Statistics (Last 30 Days)

Charger 1 **Number of Sessions:** 119 **Cumulative Session Time:** 104584

Number of Charging Sessions Per Day (Last 10 Days)

Total Duration of All Charging Sessions Per Day (Last 10 Days)

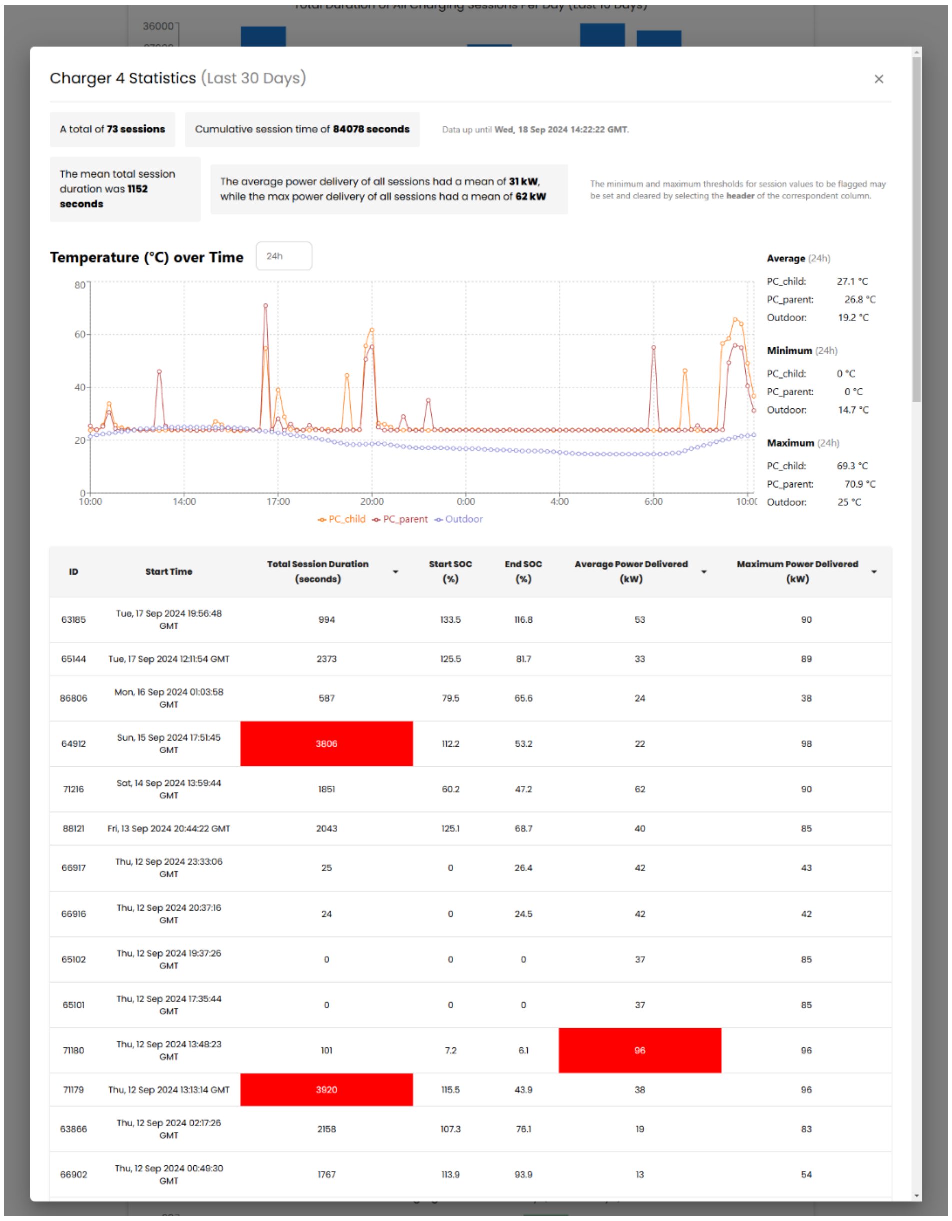
Charger 2 **Number of Sessions:** 132 **Cumulative Session Time:** 177037

Number of Charging Sessions Per Day (Last 10 Days)

Total Duration of All Charging Sessions Per Day (Last 10 Days)

Charger 3 **Number of Sessions:** 92 **Cumulative Session Time:** 116081

Figure 17. Historical record of charging sessions for each charger



Charger 4 Statistics (Last 30 Days)

A total of **73 sessions**

Cumulative session time of **84078 seconds**

Data up until **Wed, 18 Sep 2024 14:22:22 GMT.**

The mean total session duration was **1152 seconds**

The average power delivery of all sessions had a mean of **31 kW**, while the max power delivery of all sessions had a mean of **62 kW**

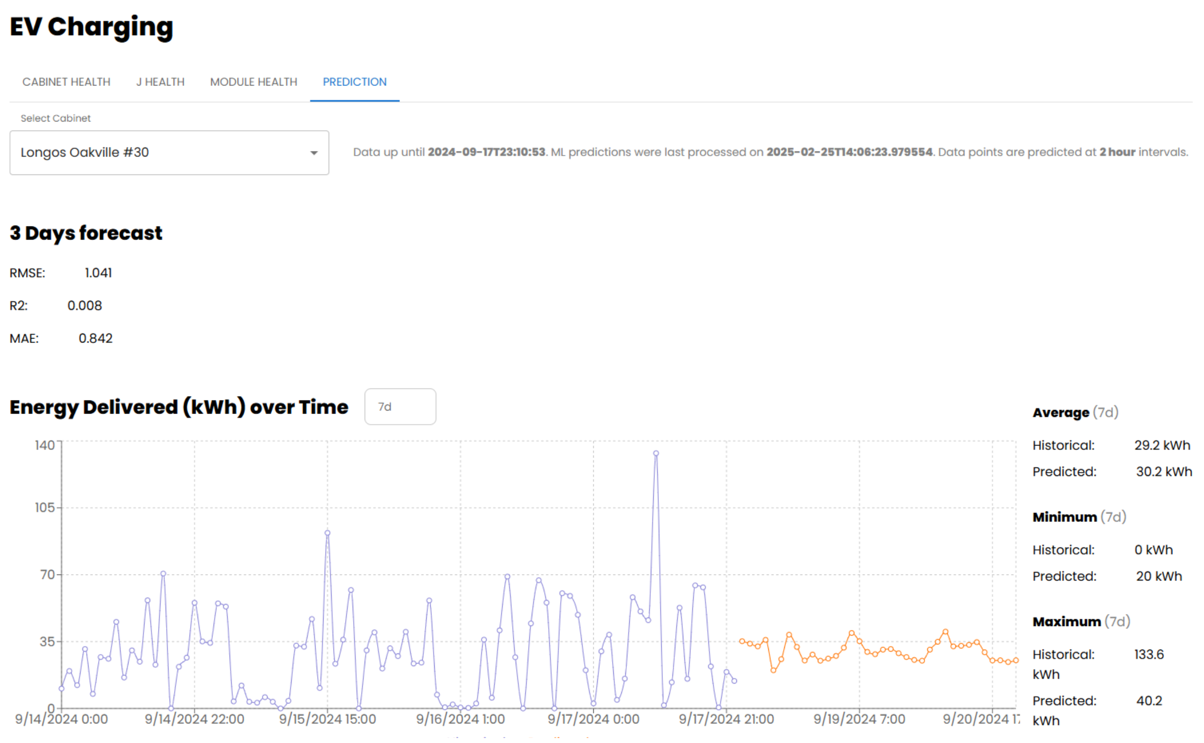
The minimum and maximum thresholds for session values to be flagged may be set and cleared by selecting the **header** of the correspondent column.

Temperature (°C) over Time

Average (24h)	
PC_child:	27.1 °C
PC_parent:	26.8 °C
Outdoor:	19.2 °C
Minimum (24h)	
PC_child:	0 °C
PC_parent:	0 °C
Outdoor:	14.7 °C
Maximum (24h)	
PC_child:	69.3 °C
PC_parent:	70.9 °C
Outdoor:	25 °C

ID	Start Time	Total Session Duration (seconds)	Start SOC (%)	End SOC (%)	Average Power Delivered (kW)	Maximum Power Delivered (kW)
63185	Tue, 17 Sep 2024 19:56:48 GMT	994	133.5	116.8	53	90
65144	Tue, 17 Sep 2024 12:11:54 GMT	2373	125.5	81.7	33	89
86806	Mon, 16 Sep 2024 01:03:58 GMT	587	79.5	65.6	24	38
64912	Sun, 15 Sep 2024 17:51:45 GMT	3806	112.2	53.2	22	98
71216	Sat, 14 Sep 2024 13:59:44 GMT	1851	60.2	47.2	62	90
88121	Fri, 13 Sep 2024 20:44:22 GMT	2043	125.1	68.7	40	85
66917	Thu, 12 Sep 2024 23:33:06 GMT	25	0	26.4	42	43
66916	Thu, 12 Sep 2024 20:37:16 GMT	24	0	24.5	42	42
65102	Thu, 12 Sep 2024 19:37:26 GMT	0	0	0	37	85
65101	Thu, 12 Sep 2024 17:35:44 GMT	0	0	0	37	85
71180	Thu, 12 Sep 2024 13:48:23 GMT	101	7.2	6.1	96	96
71179	Thu, 12 Sep 2024 13:13:14 GMT	3920	115.5	43.9	38	96
63866	Thu, 12 Sep 2024 02:17:26 GMT	2158	107.3	76.1	19	83
66902	Thu, 12 Sep 2024 00:49:30 GMT	1767	113.9	93.9	13	54

Figure 18. Charging Station battery cell health report. Abnormal sensor readings are highlighted in red



EV Charging

Select Cabinet: Longos Oakville #30

Data up until **2024-09-17T23:10:53**. ML predictions were last processed on **2025-02-25T14:06:23.979554**. Data points are predicted at **2 hour** intervals.

3 Days forecast

RMSE:	1.041
R2:	0.008
MAE:	0.842

Energy Delivered (kWh) over Time

Average (7d)

Historical:	29.2 kWh
Predicted:	30.2 kWh

Minimum (7d)

Historical:	0 kWh
Predicted:	20 kWh

Maximum (7d)

Historical:	133.6 kWh
Predicted:	40.2 kWh

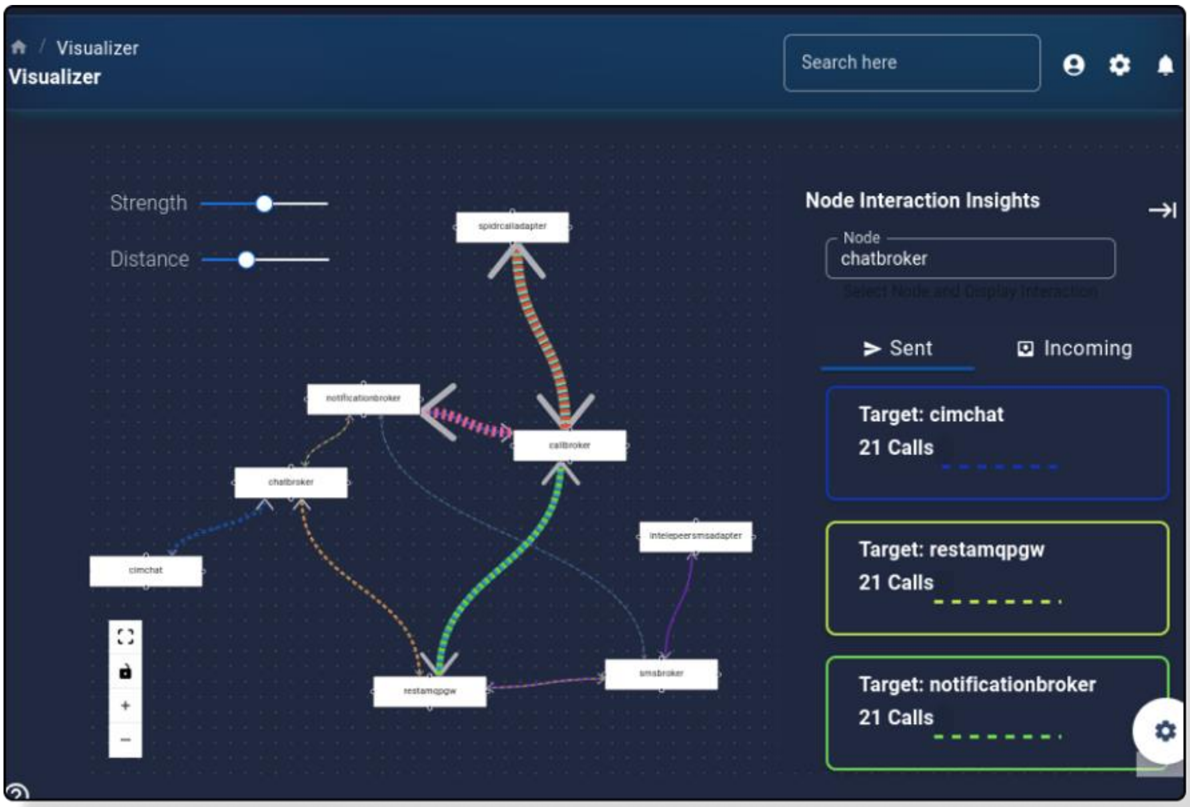
Figure 19. Charging Station energy delivery prediction

3.4. NetRD

Visualizations Used: Line chart, bar chart, radar chart, pie chart, scatter plot, tabular representation
Illustrates: Microservice dependencies and interactions over time, Identification of performance bottlenecks, Insights into how changes in architecture affect software quality
Technology Stack: DIA4M, React Flow, D3.js, Node.js
Key Features:

- Service Mapping Feature: The service mapping feature includes handling file uploads, reading the CSV file, extracting columns, filtering data, generating output with nodes and edges to be visualized and used as input in other analytics related modules.
- Anomaly Detection Feature: Anomaly detection is an important aspect of system monitoring, especially to identify irregularities in log data that may indicate possible system failures or security breaches. This process uses a mixture of statistical methods and domain-specific heuristics to improve detection accuracy.
- Service Health Feature: The Services Health Summary UI in DIA4M redefines the way DevOps engineers analyse microservice performance through a visually engaging table with embedded line graphs. This interface provides a comprehensive view of critical metrics for each service.
- Logs Comparison Feature: The Log Comparison feature facilitates the analysis of existing logs in relation to those in previous versions and enables users to discern changes and patterns in log data. Using the "Horizontal Comparison" method, the tool clusters messages based on their instances in each log file and then compares these clusters, highlighting differences and similarities. This approach allows for a detailed examination of log data and supports selective analysis of specific log areas of interest, increasing the ability to effectively understand and monitor log changes.

Dashboard:



Node Interaction Insights

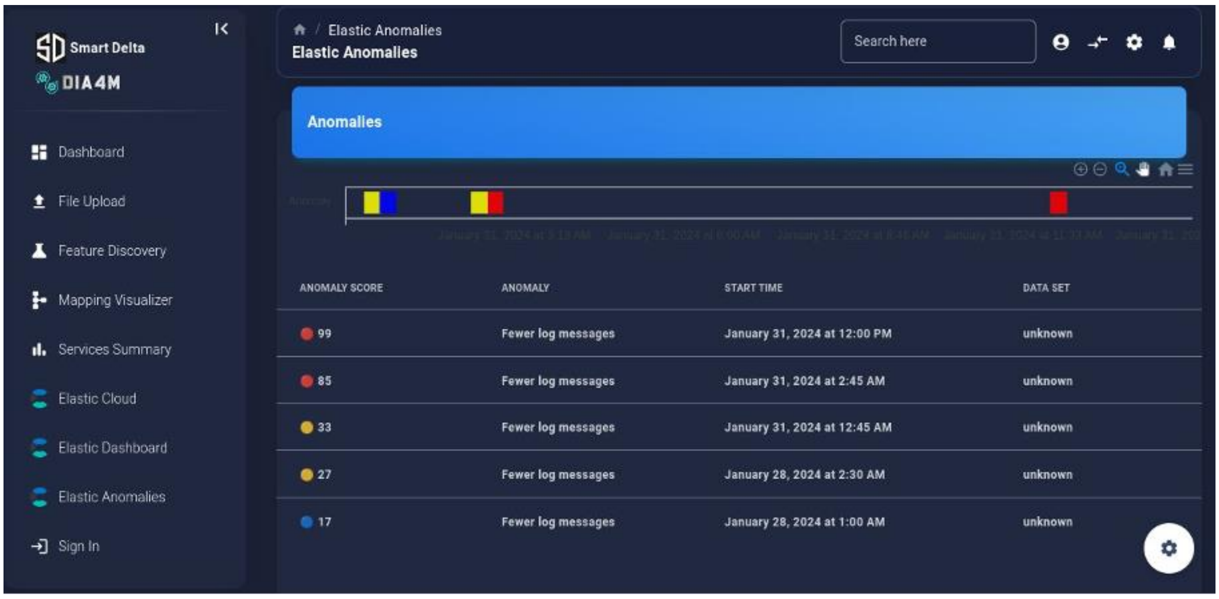
Node: chatbroker

Target: cimchat
21 Calls

Target: restamqpgw
21 Calls

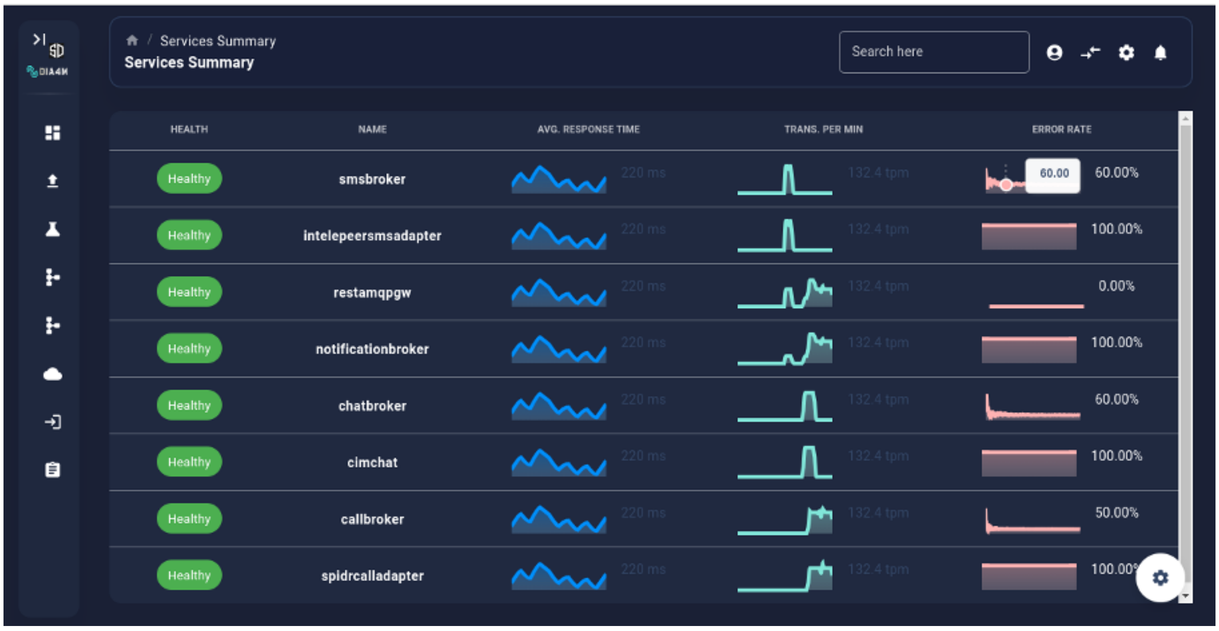
Target: notificationbroker
21 Calls

Figure 20. Service mapping by log file



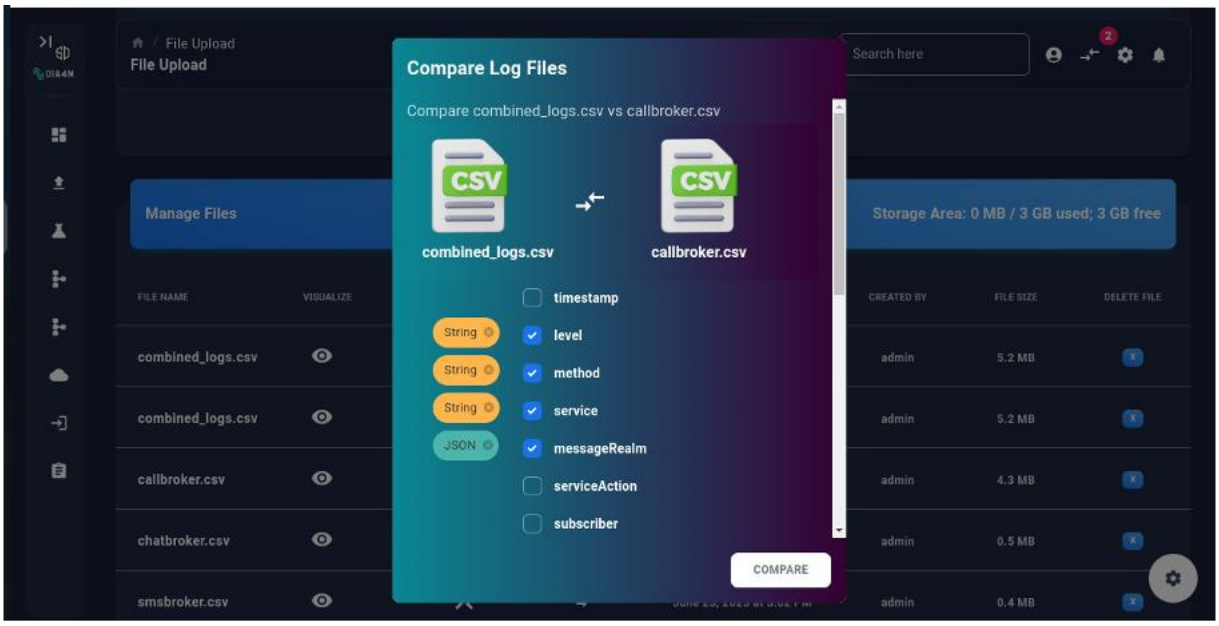
ANOMALY SCORE	ANOMALY	START TIME	DATA SET
🔴 99	Fewer log messages	January 31, 2024 at 12:00 PM	unknown
🔴 85	Fewer log messages	January 31, 2024 at 2:45 AM	unknown
🟡 33	Fewer log messages	January 31, 2024 at 12:45 AM	unknown
🟡 27	Fewer log messages	January 28, 2024 at 2:30 AM	unknown
🔵 17	Fewer log messages	January 28, 2024 at 1:00 AM	unknown

Figure 21. Anomaly detection with scoring


Services Summary

HEALTH	NAME	AVG. RESPONSE TIME	TRANS. PER MIN	ERROR RATE
Healthy	smsbroker	220 ms	132.4 tpm	60.00%
Healthy	intelepeersmsadapter	220 ms	132.4 tpm	100.00%
Healthy	restamqpgw	220 ms	132.4 tpm	0.00%
Healthy	notificationbroker	220 ms	132.4 tpm	100.00%
Healthy	chatbroker	220 ms	132.4 tpm	60.00%
Healthy	cimchat	220 ms	132.4 tpm	100.00%
Healthy	callbroker	220 ms	132.4 tpm	50.00%
Healthy	spidrcalladapter	220 ms	132.4 tpm	100.00%

Figure 22. Each service`s health status in a single view


File Upload

Compare Log Files

Compare combined_logs.csv vs callbroker.csv

combined_logs.csv — callbroker.csv

☐	timestamp
String ☑	level
String ☑	method
String ☑	service
JSON ☑	messageRealm
☐	serviceAction
☐	subscriber

COMPARE

Manage Files — Storage Area: 0 MB / 3 GB used; 3 GB free

FILE NAME	CREATED BY	FILE SIZE
combined_logs.csv	admin	5.2 MB
combined_logs.csv	admin	5.2 MB
callbroker.csv	admin	4.3 MB
chatbroker.csv	admin	0.5 MB
smsbroker.csv	admin	0.4 MB

Figure 23. Logs comparison feature

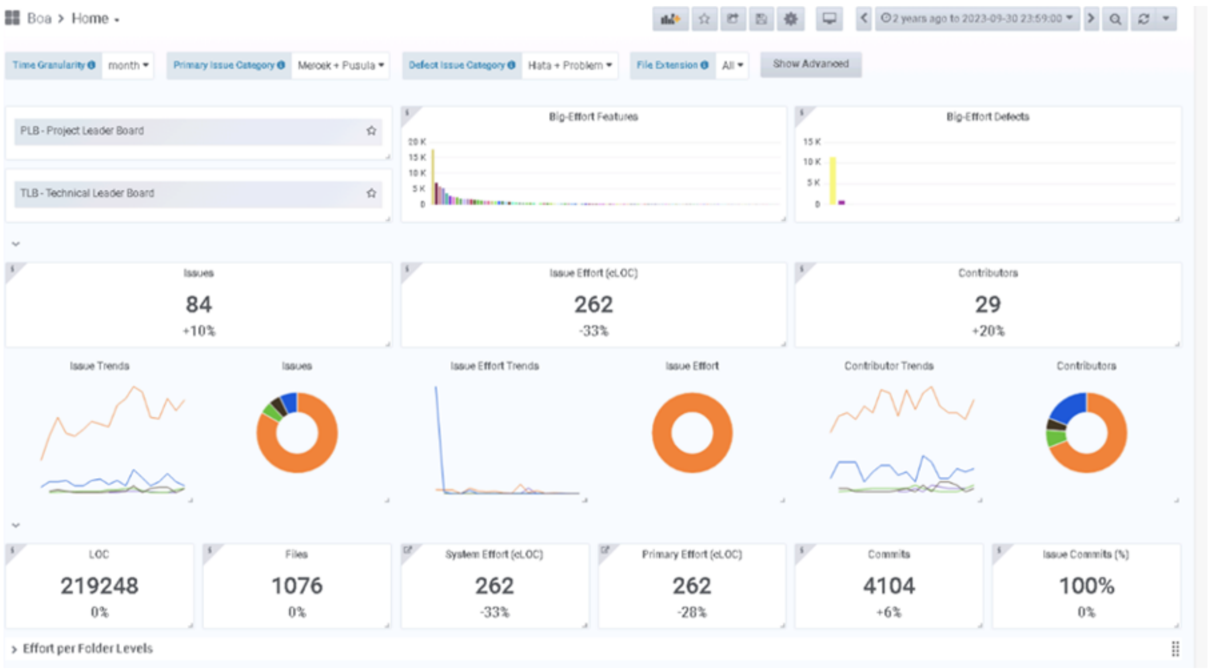
3.5. Kuveyt Türk

Visualizations Used: Network diagram, bubble chart, line chart, pie chart, radar chart, timeseries chart.
Illustrates: Code quality metrics (code quality, code smell, code coverage, technical debt, delta analysis), performance metrics, maintainability metrics (knowledge distribution, knowledge islands, dependency & coupling metrics, architecture maintainability), deployment and process metrics (bug resolution time, feature vs. bug distribution)
Technology Stack: Detangle, Grafana
Key Features:

- Establish correlation between production performance metrics and software quality metrics.
- Analyse, locate and measure Technical Debt and knowledge distribution issues.
- Track trends in architectural maintainability and software extensibility.
- Identify key contributors, knowledge silos, and potential risks in development teams.

- Provide time-series analysis of software quality and deployment impact over multiple versions.

Dashboard:



Time Granularity: month | Primary Issue Category: Merek + Pusula | Defect Issue Category: Hata + Problem | File Extension: All

Issues 84 +10%

Issue Effort (eLOC) 262 -33%

Contributors 29 +20%

LOC 219248 0%

Files 1076 0%

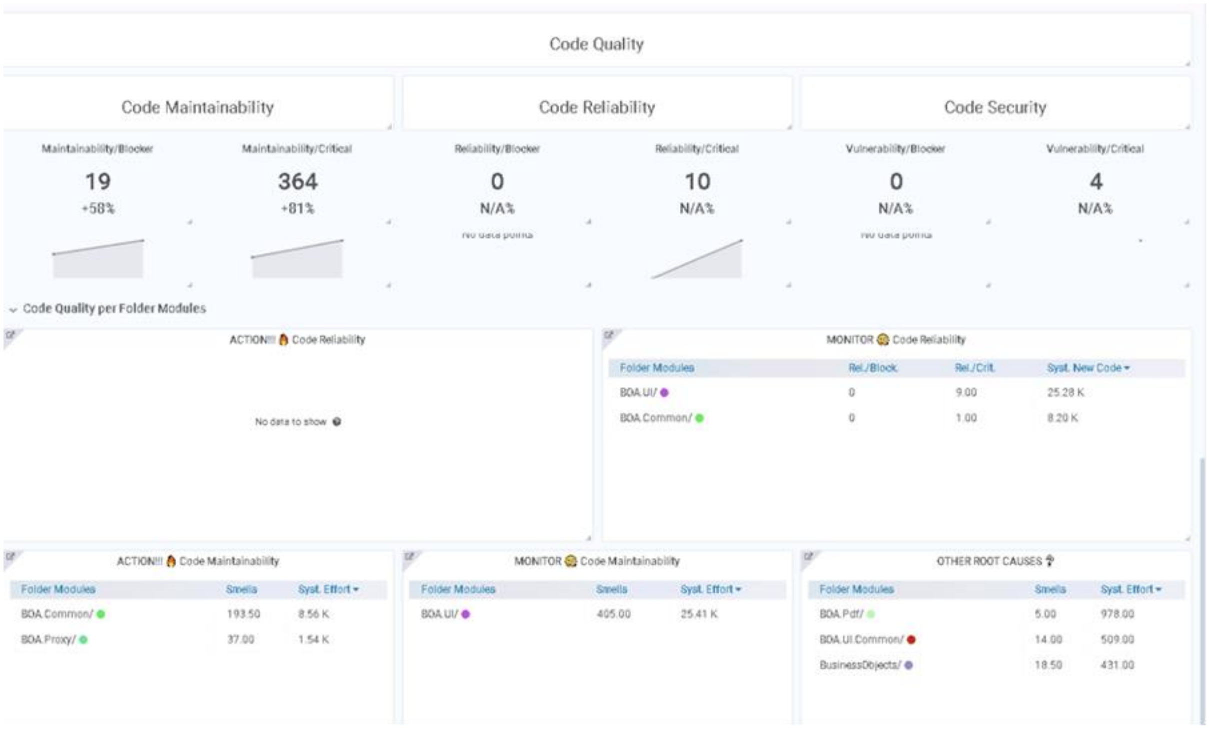
System Effort (eLOC) 262 -33%

Primary Effort (eLOC) 262 -28%

Commits 4104 +6%

Issue Commits (%) 100% 0%

Figure 24. Breakdown of the features and bugs



Maintainability/Blocker 19 +58%

Maintainability/Critical 364 +81%

Reliability/Blocker 0 N/A%

Reliability/Critical 10 N/A%

Vulnerability/Blocker 0 N/A%

Vulnerability/Critical 4 N/A%

ACTION!!! Code Reliability

No data to show

MONITOR Code Reliability

Folder Modules	Rel./Block	Rel./Crit.	Syst. New Code
BOA.UI/ ●	0	9.00	25.28 K
BOA.Common/ ●	0	1.00	8.20 K

ACTION!!! Code Maintainability

Folder Modules	Smells	Syst. Effort
BOA.Common/ ●	193.50	8.56 K
BOA.Proxy/ ●	37.00	1.54 K

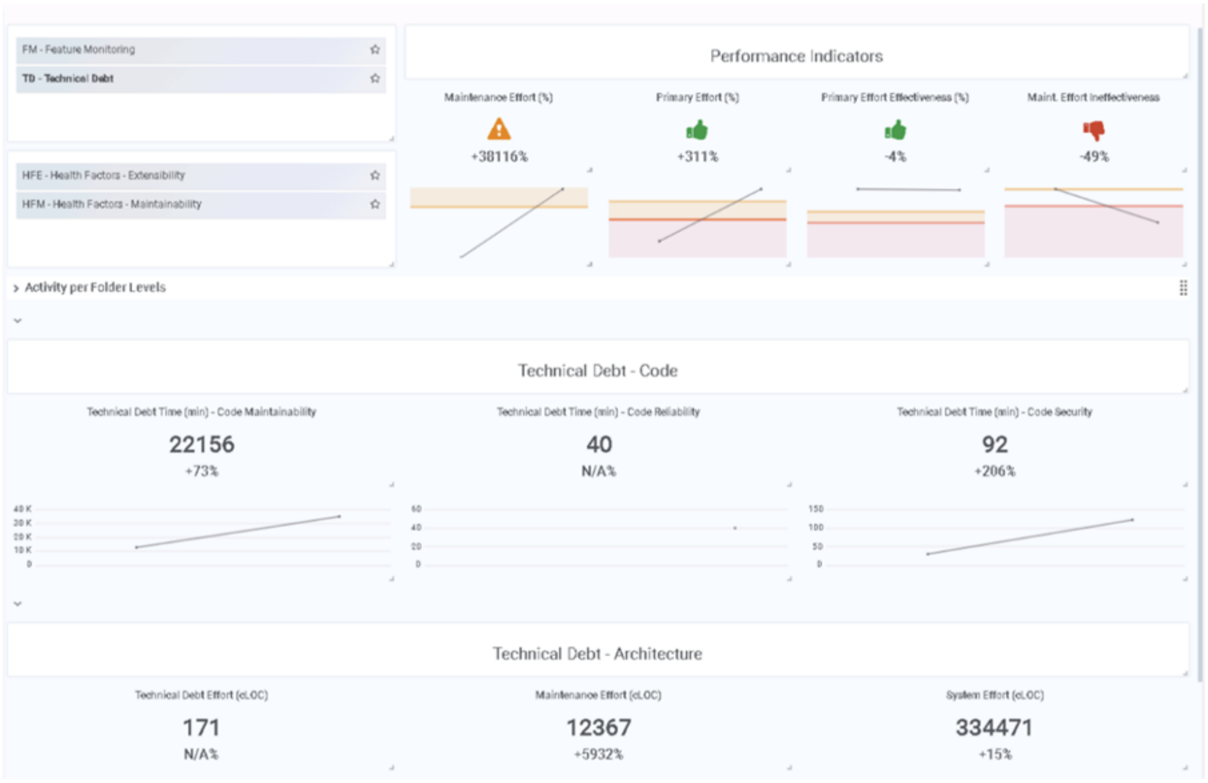
MONITOR Code Maintainability

Folder Modules	Smells	Syst. Effort
BOA.UI/ ●	405.00	25.41 K

OTHER ROOT CAUSES

Folder Modules	Smells	Syst. Effort
BOA.Pdf/ ●	5.00	978.00
BOA.UI.Common/ ●	14.00	509.00
BusinessObjects/ ●	18.50	431.00

Figure 25. Static code quality metrics


FM - Feature Monitoring

TD - Technical Debt

HFE - Health Factors - Extensibility

HFM - Health Factors - Maintainability

Performance Indicators

Maintenance Effort (%)	Primary Effort (%)	Primary Effort Effectiveness (%)	Maint. Effort Ineffectiveness
+38116%	+311%	-4%	-49%

Activity per Folder Levels

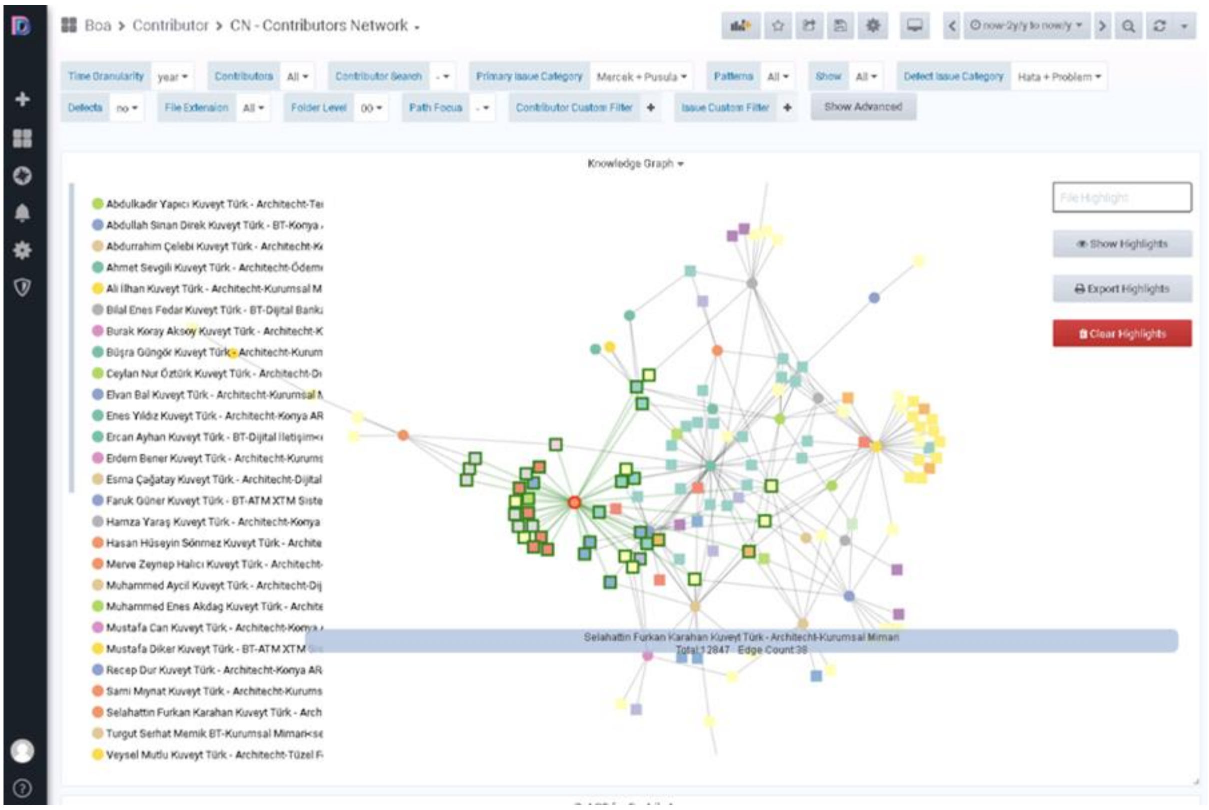
Technical Debt - Code

Technical Debt Time (min) - Code Maintainability	Technical Debt Time (min) - Code Reliability	Technical Debt Time (min) - Code Security
22156	40	92
+73%	N/A%	+206%

Technical Debt - Architecture

Technical Debt Effort (cLOC)	Maintenance Effort (cLOC)	System Effort (cLOC)
171	12367	334471
N/A%	+5932%	+15%

Figure 26. Technical debt metrics


Boa > Contributor > CN - Contributors Network

now-2y/y to now/y

Time Granularity: year; Contributors: All; Contributor Search: -; Primary Issue Category: Mercek + Pusula; Patterns: All; Show: All; Defect Issue Category: Hata + Problem

Defects: no; File Extension: All; Folder Level: 00; Path Focus: -; Contributor Custom Filter; Issue Custom Filter; Show Advanced

Knowledge Graph

Abdulkadir Yapıcı Kuveyt Türk - Architecht-Te
Abdullah Sinan Direk Kuveyt Türk - BT-Konya
Abdurrahim Çelebi Kuveyt Türk - Architecht-K
Ahmet Sevgili Kuveyt Türk - Architecht-Ödem
Ali İlhan Kuveyt Türk - Architecht-Kurumsal M
Bilal Enes Fedar Kuveyt Türk - BT-Dijital Bank
Burak Koray Aksoy Kuveyt Türk - Architecht-K
Büşra Güngör Kuveyt Türk - Architecht-Kurum
Ceylan Nur Öztürk Kuveyt Türk - Architecht-D
Elvan Bal Kuveyt Türk - Architecht-Kurumsal
Enes Yıldız Kuveyt Türk - Architecht-Konya AR
Ercan Ayhan Kuveyt Türk - BT-Dijital İletişim
Erdem Bener Kuveyt Türk - Architecht-Kurums
Esma Çağatay Kuveyt Türk - Architecht-Dijital
Faruk Güner Kuveyt Türk - BT-ATM XTM Siste
Hamza Yaraş Kuveyt Türk - Architecht-Konya
Hasan Hüseyin Sönmez Kuveyt Türk - Archite
Merve Zeynep Halıcı Kuveyt Türk - Architecht-
Muhammed Aycil Kuveyt Türk - Architecht-Dij
Muhammed Enes Akdag Kuveyt Türk - Archite
Mustafa Can Kuveyt Türk - Architecht-Konya
Mustafa Diker Kuveyt Türk - BT-ATM XTM
Recep Dur Kuveyt Türk - Architecht-Konya AR
Sami Mıynat Kuveyt Türk - Architecht-Kurums
Selahattin Furkan Karahan Kuveyt Türk - Arch
Turgut Serhat Memik BT-Kurumsal Mimarı
Veysel Mutlu Kuveyt Türk - Architecht-Tüzel F
File Highlight
Show Highlights
Export Highlights
Clear Highlights
Selahattin Furkan Karahan Kuveyt Türk - Architecht-Kurumsal Miman
Total 12847 Edge Count 38

Figure 27. Developer analysis

3.6. GlassHouse

Visualizations Used: Time series chart, area curve, scatter lot, line graph.

Illustrates: Key performance indicators such as Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR), incident response times, threat prioritization, anomaly detection
Technology Stack: Python, Flux
Key Features:

- **Centralized Dashboard:**
  - **Real-time Monitoring:** Instantaneous updates that allow teams to monitor KPIs as they evolve.
  - **Customizable Views:** Users can tailor the dashboard to display the most relevant metrics and alerts for their specific role or operational focus.
  - **Interactive Drill-Down:** Beyond high-level overviews, users can click into specific charts or metrics to see more granular data, helping in root-cause analysis.
  - **Alerting & Notifications:** With integration to ServiceNow and other systems, the dashboard can trigger alerts based on pre-defined thresholds or detected anomalies.
  - quickly identify inefficiencies, optimize incident response processes, and prioritize threats effectively.

Dashboard:

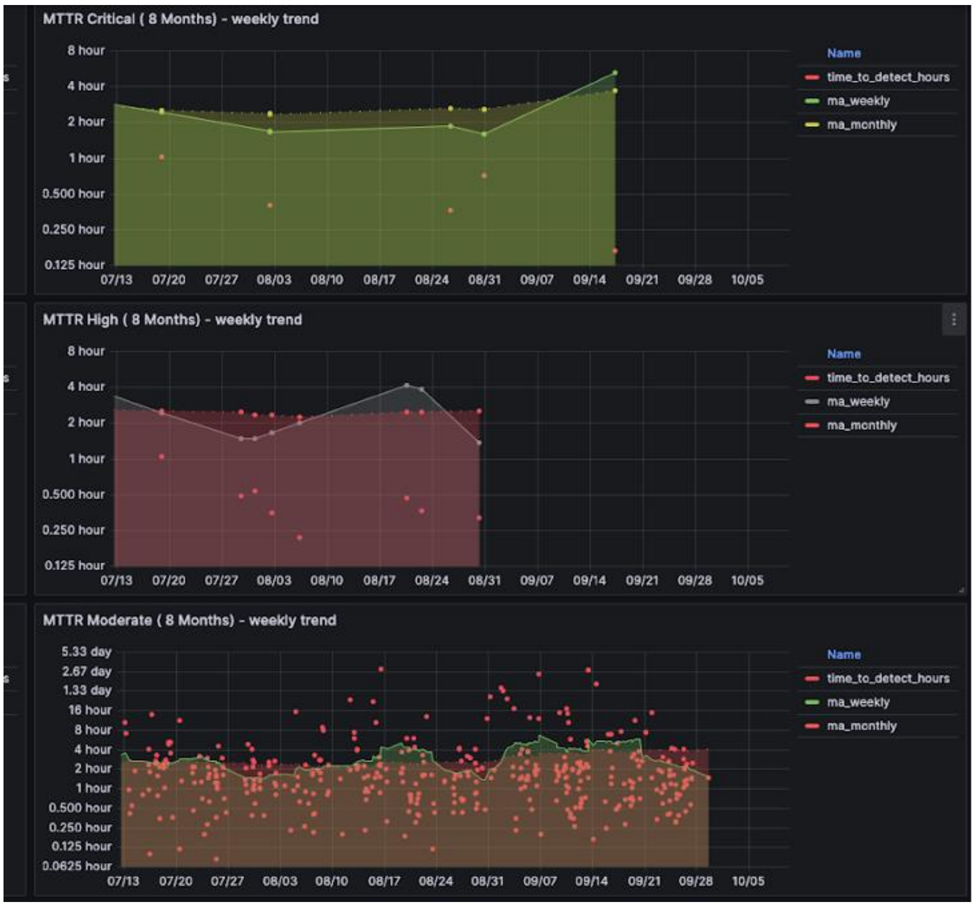


Figure 28. Grafana dashboards to analyse key KPI metrics

3.7. Team Eagle

Visualizations Used: Line graph, histogram, scatter plot, pie chart
Illustrates: Software quality assurance and compliance to structured coding conventions
Technology Stack: JavaScript libraries (MaterialUI, Axios and ChartJS), NodeJS, ReactJS, Python Flask, Azure Repos, Azure DevOps.
Key Features:

- Display potential SQA-related issues found within the codebase
- Identifies issue types, vulnerabilities, severities, and possible resolutions
- Visualizations of system metrics, fault detection, and predictive insights
- Integration with event-driven data processing pipelines for daily monitoring

Dashboard:

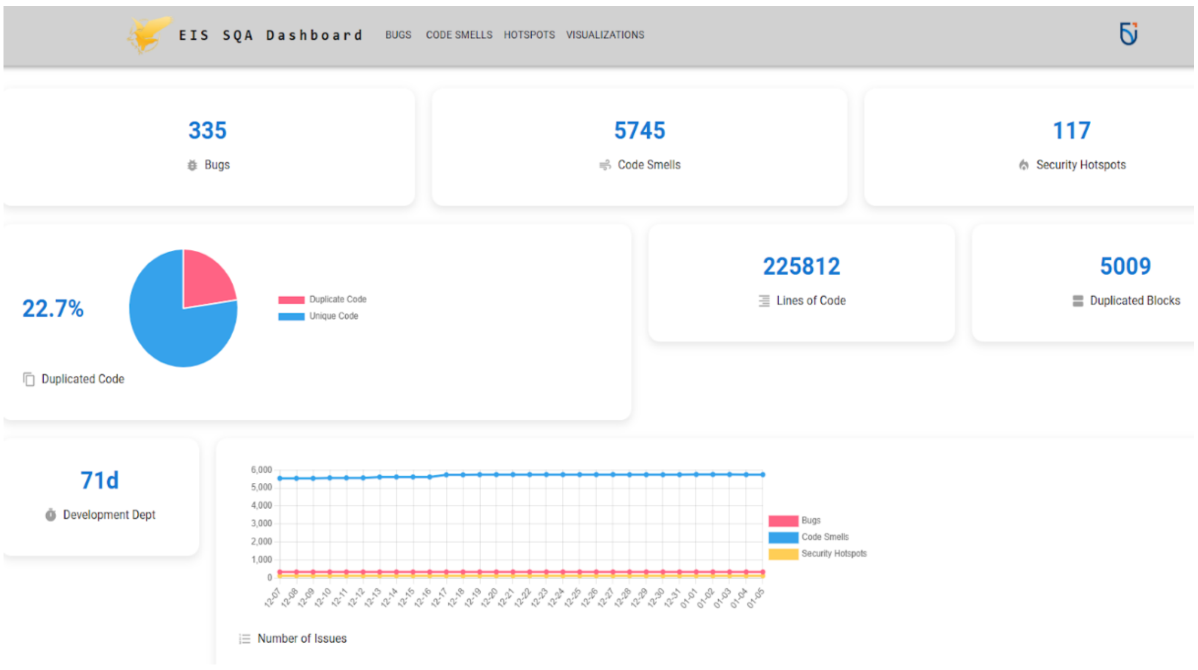


Figure 29. Team Eagle QA Dashboard Homepage

3.8. Arçelik

Visualizations Used: Pie charts, line charts, radar charts,
Illustrates: Product quality metrics, code smells, vulnerabilities, technical debt, process-oriented metrics, and test coverage
Technology Stack: Qlik
Key Features:

- The metric dashboard integrates data from:
  - Azure DevOps
  - SonarQube
  - Jira
  - Statuspage
  - Google Analytics
- Provides a comprehensive view of IT software processes and quality metrics
- Tracks key indicators:
  - DevSec percentage (ATRT)
  - PR cycle times
  - Code quality
  - Incident response
  - Deployment frequency
- Enables data-driven decision-making by:
  - Visualizing trends
  - Identifying bottlenecks

- Ensuring continuous improvement in software development and operations

Dashboard:

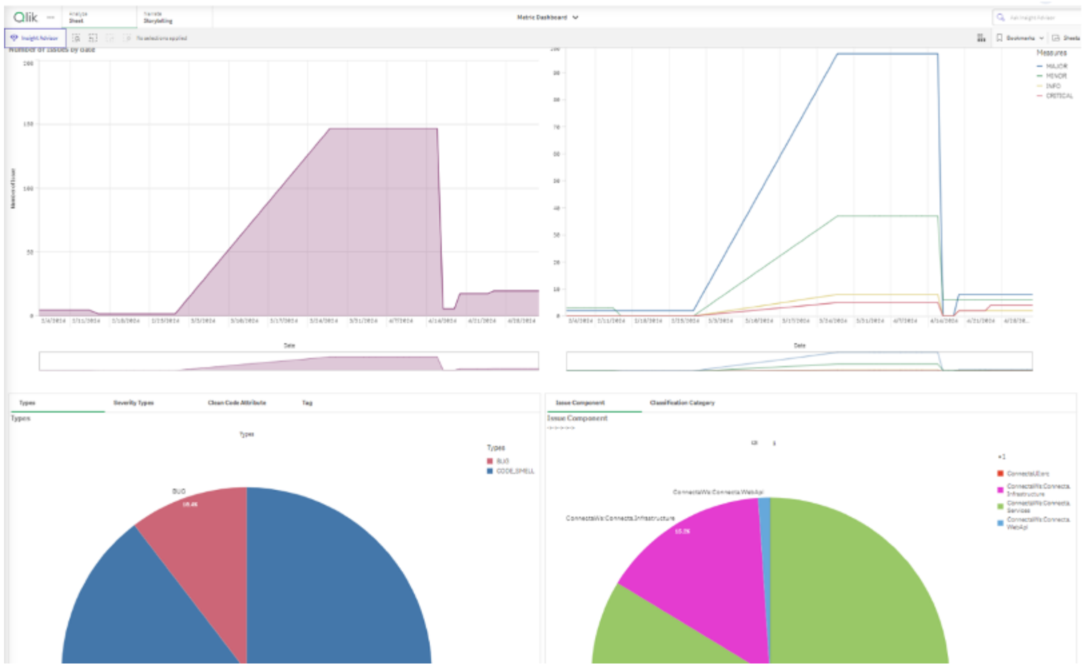


Figure 30. Metric Dashboard: The issues linked with the related pull requests

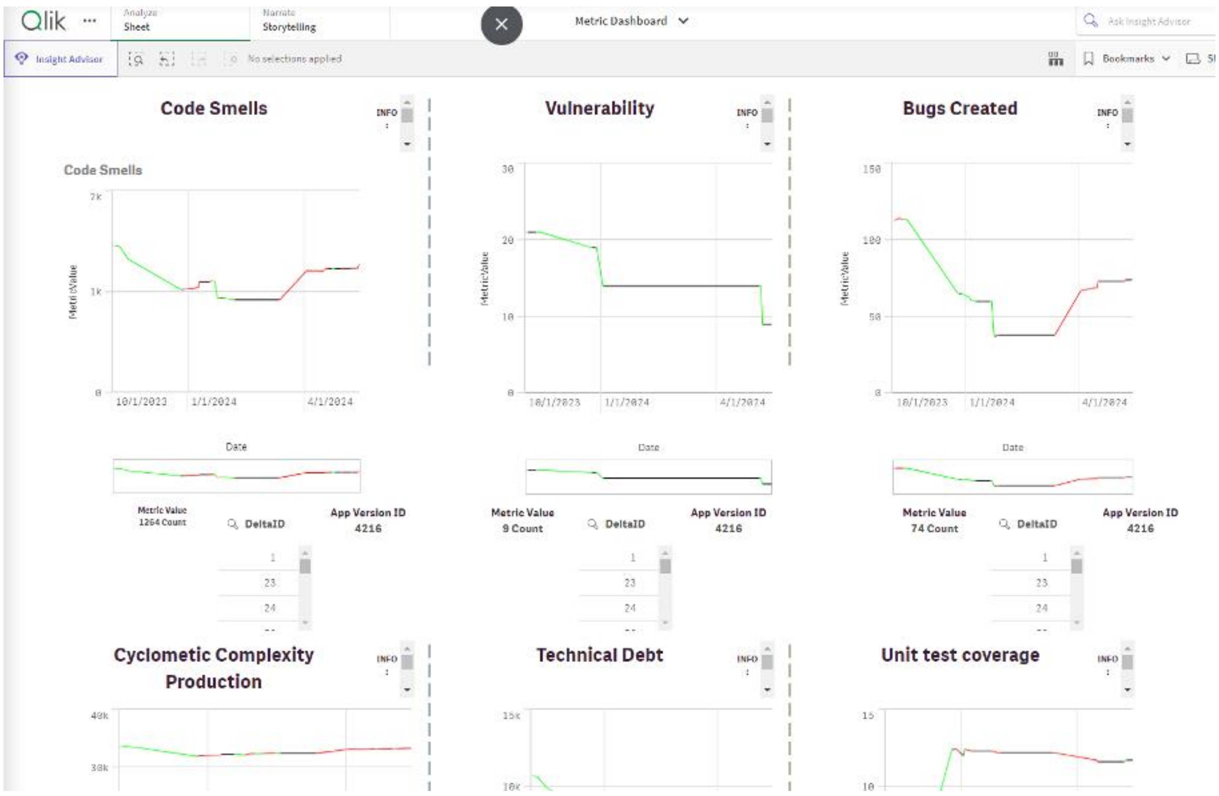


Figure 31. Metric Dashboard: Product Metrics

4. Tools

4.1. Detangle (Cape of Good Code)

DETANGLE analyzes the effectiveness of software development strategies, detecting the efficiency of individual software developers and software teams and the evolution of their products over the years to provide detailed and dynamic reporting. With these reports, customers will be able to understand the importance of their developers in a particular module or a whole project. With its creative feature-based effort and quality analysis, DETANGLE reveals unbalances between R&D effort and the commercial worth of features. Software development and business may work together taking into consideration the DETANGLE analysis and heated debates to become data-driven, fact-based business judgments.

Input to the tool: Project Repository and Issue Tracker

Output of the tool: Software Quality Metrics

Webpage: https://capeofgoodcode.com/

Version: Released (Closed source)

Contact person: Egon Wuchner (wuchner@capeofgoodcode.com)

4.2. SoHist (University of Innsbruck, c.c.com)

Technical debt is often the outcome of short-term judgments taken during code creation, which can result in long-term maintenance costs and risks. In this approach, assessing the project's development and comprehending various effect factors is critical. Fortunately, code analysis tools such as the well-known SonarQube can assist in the prioritizing process for reducing technical debt. Therefore, we introduced two new visualization approaches:

Given SonarQube's industrial relevance in 2023 and the demand for extended (historical) analysis functionality, we decided to create SoHist. This new tool addresses SonarQube's limitations and provides extended historical code analysis capabilities, such as the evolution of TD over time.

After the execution of SonarQube analysis, the user can access the code quality history and use the two available SoHist visualizations – *Code Evolution* and *Weighted Code Evolution Significance.*

- Visualization 1- Code Evolution: SoHist enables simultaneous viewing of multiple code quality metrics. When the user hovers over a specific time, the corresponding timestamp is highlighted across all metric charts. This facilitates easy comparison of the metrics.

- Visualization 2 - Weighted Code Evolution Significance: This visualization introduces a novel approach to address the challenge of individual project demands and prioritization of specific SonarQube's main metrics. The Weighted Code Evolution Significance serves as an indicator of the significance of changes in relation to weighted categories.

Input to the tool: Credentials to GitLab Repository
Output of the tool: Complete GIT-History Quality Assurance Analysis with SonarQube and Visualization for Interpretation of Technical Debt

Webpage:

- Publication on ITEA website: https://itea4.org/news/smartdelta-offers-solution-for-retro-perspective-code-analysis-in-technical-debt-management.html

- Paper: https://www.researchgate.net/publication/370489799_SoHist_A_Tool_for_Managing_Technical_Debt_through_Retro_Perspective_Code_Analysis
- GitHub Repository: https://github.com/bdornauer/sohist

Version: v2
Source or Binary Link: https://github.com/bdornauer/sohist
Instruction manual or tutorial for the tool:
https://github.com/bdornauer/sohist/blob/main/README.md
Type: *OpenSource*
Contact person: benedikt.dornauer@uibk.ac.at

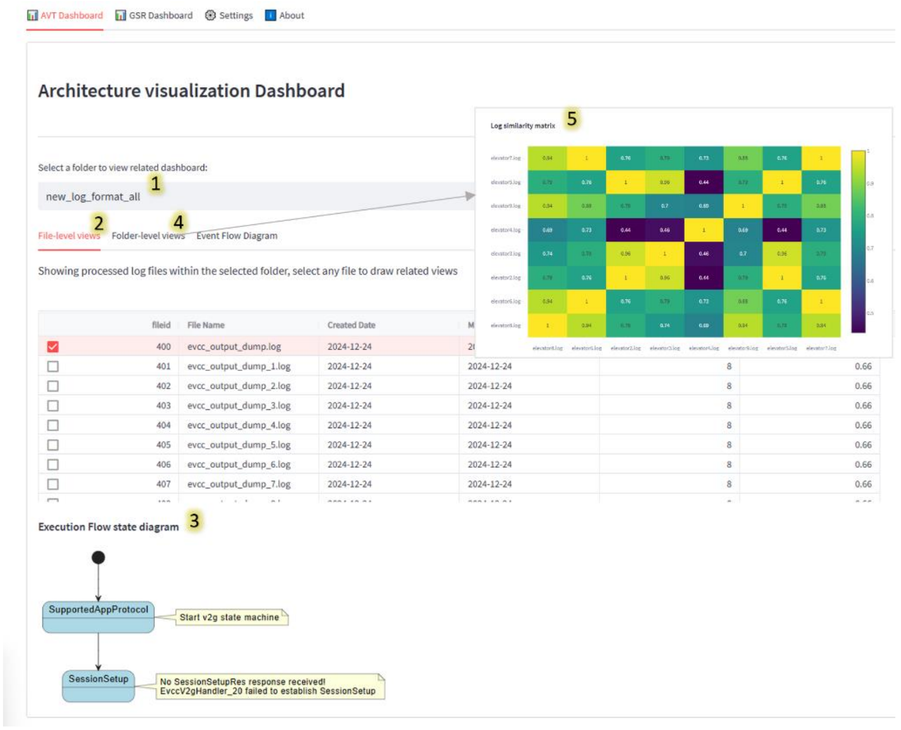
4.3. Architecture Analysis and Visualization Tool (Fraunhofer)

The architecture visualization tool computes various architectural views based on the input files. The data analysis module of the tool takes a set of log files that capture data from state machine executions as input and analyzes them to compute various architectural diagrams. These include:

- An **Execution Flow State Diagram** that captures the runtime behaviour of a model based on the log file data
- A **Log Similarity Matrix** which is a heatmap that shows pair-wise similarity between each pair of ingested log files.

These views are computed in a textual format (markdown-like syntax and dataframes) and are stored in a database.

A graphical representation of the computed diagrams is then available through a web-based visualization dashboard. As can be seen in the figure below, the dashboard provides options to select processed folders (1) and files within those folders through a grid-view (2). Selection of files then generates the corresponding view (3). The similarity matrix (5) is generated at folder-level and is available from the "folder-level views" tab (4).



Architecture visualization Dashboard

Select a folder to view related dashboard:

new_log_format_all

Showing processed log files within the selected folder, select any file to draw related views

	fileid	File Name	Created Date	M		
✓	400	evcc_output_dump.log	2024-12-24	2		
☐	401	evcc_output_dump_1.log	2024-12-24	2024-12-24	8	0.66
☐	402	evcc_output_dump_2.log	2024-12-24	2024-12-24	8	0.66
☐	403	evcc_output_dump_3.log	2024-12-24	2024-12-24	8	0.66
☐	404	evcc_output_dump_4.log	2024-12-24	2024-12-24	8	0.66
☐	405	evcc_output_dump_5.log	2024-12-24	2024-12-24	8	0.66
☐	406	evcc_output_dump_6.log	2024-12-24	2024-12-24	8	0.66
☐	407	evcc_output_dump_7.log	2024-12-24	2024-12-24	8	0.66

Execution Flow state diagram

Figure 32. Visualization dashboard showing file selection grid with file-level and folder-level views.

Additionally, the dashboard provides functionality to construct an **Event Flow Diagram** (6), illustrating the relationships among state machines based on shared events. Users can drag and drop multiple .ceps files, which define state machines, into the input widget in the UI. These files are then processed to extract both outgoing and incoming events associated with the state machines defined by each .ceps file. The resulting diagram displays state machines as nodes and uses event connections as edges.

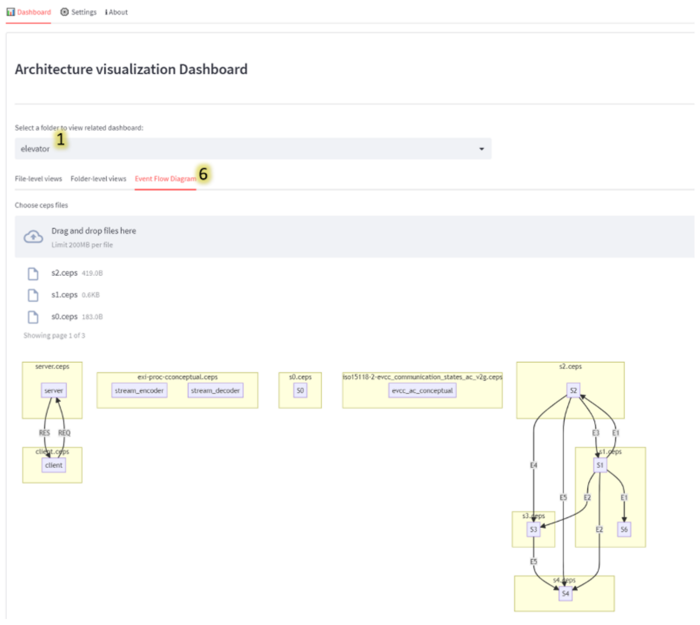


Figure 33. Architecture Visualization Dashboard – Execution Flow State Diagram.

These diagrams are helpful in visualizing the run-time behaviour of a state machine and aid in fault diagnosis. For example, when a fault occurs, the corresponding log file can be analyzed, allowing users to observe the sequence of events graphically via the execution flow state diagram. Further, the similarity matrix facilitates the identification of similar logs from previous runs, aiding in the discovery of similar faults. As a result, these diagrams provide valuable insights for initial quality control, effectively localizing faults and supporting the diagnostic process.

Input to the tool: Folder containing log files.

Output of the tool: Sequence diagrams and heatmaps at file and folder level.

Version: 0.2.0

Source or Binary Link:
https://github.com/SmartDeltaFraunhoferFOKUS/Architecture_Visualization_Tool

Instruction manual or tutorial for the tool:
https://github.com/SmartDeltaFraunhoferFOKUS/Architecture_Visualization_Tool/wiki

Additional information:

Contact person: abhishek.shrestha@fokus.fraunhofer.de

4.4. EPS Cybersecurity Anomaly Detector (Glasshouse Systems and Ontario Tech University)

The Internet is a cyber world consisting of more data than humanly imaginable. Every time someone accesses the Internet from one of their devices, they create a detailed trail of events from the moment they begin until the moment they stop. These events are essential to understand what is happening in a network. Keeping track of millions of events is not a process that can be done manually but instead requires the help of software known as Security Information and Event Management (SIEM) solutions. SIEM solutions have been a game changer in the past two decades. However, as the amount of data being generated daily grows exponentially, the current configuration of SIEM solutions falls behind in capability. A growing trend in software solutions is the use of artificial intelligence, specifically machine learning to help make processes more efficient. This trend is starting to be adopted in the cybersecurity space but still requires more innovation. This tool utilises querying and data processing techniques that feed into an unsupervised model and can help an analyst quickly detect anomalies that other measures would have otherwise missed.

Input to the tool: EPS information

Output of the tool: Anomalies, visualization on detected anomalies, resource usage visualization

Version: V1.0

Type: Closed

Additional information:

Contact person: jgardiner@ghsystems.com, agil@ghsystems.com

4.5. DRACONIS (Mälardalen University and Alstom)

DRACONIS acts as a static analysis framework for block-based development models used in the context of quality assurance. The primary use of the tool is to check the quality and extract metrics from 61131-3 Function Block Diagrams. DRACONIS can also compute a semantic and graphical delta report between multiple versions of models. DRACONIS comes prepackage with both a command-line interface and a web application, allowing different stakeholders to analyse, review and generate reports for models. It supports external metrics-based tools through offline analysis and upload to the web application.

Input to the tool: Function-block Diagrams in POU format
Output of the tool: Reports – text and Excel
Webpage & Instruction manual for the tool: https://github.com/jean-malm-mdh/draconis
Version: 0.4
Contact Person: Jean Malm (jean.malm@mdu.se)

5. Conclusion

SmartDelta Visualization Dashboard represents a crucial milestone in the project, delivering a powerful and flexible platform for analyzing and optimizing software quality. Through its centralized Vaadin-based dashboard, stakeholders can access a unified interface that presents diverse metrics and visualizations from different use cases within the SmartDelta ecosystem. By leveraging the Vaadin charts library, the dashboard enables the seamless integration and visualization of various data sources, facilitating a comprehensive understanding of software evolution and quality characteristics.

Moreover, this deliverable reports on the individually developed dashboards contributed by partners, each tailored to specific analytical requirements. These partner-contributed visualizations, ranging from code quality dashboards to resource utilization analysis and anomaly detection, demonstrate the versatility and extensibility of the SmartDelta Visualization Dashboard. Presented with these specialized dashboards, stakeholders gain access to a rich software analytics environment that fosters collaboration, data-driven decision-making, and continuous improvement efforts.