

CodeScene.io

Software Evolution and Reverse Engineering

Refolli F. 865955

March 2, 2025

Projects Overview

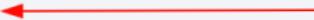
3 projects

[+ Add new project](#)

Active filters:

Status: All

Sorting: Name (rever...)



Search by project name

List Graph



Software Portfolio

Total projects: 3

Projects overview by Code Health: Healthy: 1 Attention: 2 Risky: 0

[Expand Portfolio](#)

spotbugs

Last analysis: Nov 20, 2024, 7:48:42 PM

[Dashboard →](#)

Code Health Attention ↗ Improving

Knowledge Distribution Healthy No change

Team-Code Alignment N/A No change

Delivery N/A No change

Demo-MongoDB

⚠ New analysis attempt failed! Last analysis: Sep 6, 2024, 1:37:59 PM

[Dashboard →](#)

Code Health Attention ↗ Improving

Knowledge Distribution Attention ↘ Declining

Team-Code Alignment Attention ↘ Declining

Delivery Attention ↗ Improving

Projects Overview



Software Portfolio

Total projects: 3

Projects overview by Code Health:

Healthy: 1 Attention: 2 Risky: 0

[Collapse Portfolio](#)

Size

Knowledge Distribution

Team-Code Alignment

Delivery

Code Coverage



Projects Overview

 **Software Portfolio** ⓘ Total projects: 3 | Projects overview by Code Health: Healthy: 1 Attention: 2 Risky: 0 [Expand Portfolio](#) ▾

spotbugs ★ ⋮ [Dashboard](#) ➔
Last analysis: Nov 20, 2024, 7:48:42 PM

Code Health



MONTHLY TRENDS

Metric	Status	Notes
Code Health	Attention ⓘ	Improving
Knowledge Distribution	Healthy ⓘ	No change
Team-Code Alignment	N/A ⓘ	No change
Delivery	N/A ⓘ	No change

Code Health ⓘ Attention ⓘ Improving

Knowledge Distribution ⓘ Healthy ⓘ No change

Team-Code Alignment ⓘ N/A ⓘ No change

Delivery ⓘ N/A ⓘ No change

MONTHLY TRENDS



21/10 28/10 4/11 11/11 18/11

Define teams Configure PM integration

Additional configuration steps are required before this factor can be used.

Additional configuration steps are required before this factor can be used.

Spotbugs / Dashboard

CodeScene

BACK TO PROJECTS

SPOTBUGS

Dashboard

Scope

Goals

Code

Architecture

Team Dynamics

System

Delivery Performance

Simulations

Configuration

GLOBAL

Configure

Learn

frefolfi

spotbugs

Lines of code 275,824

Programming languages

Code Health

Year reference

Attention

Knowledge Distribution

Healthy

Team-Code Alignment

Not available

Delivery

Not available

Define teams

Configure PM integration

What is Code Health?

Hotspot Code Health 3.9 (Unhealthy)

Average Code Health 8 (Pristine)

Worst Performer 1 (Unhealthy)

View hotspots

Explore codebase

View worst performers

Filter

TW 1M 1Y

What is Code Health?

Hotspots are the files with most development activity. Even a minor amount of technical debt in a hotspot will become expensive due to the high development activity.

View hotspots

Average Code Health

The weighted average health of all the files in the codebase. This metric indicates how deep any potential code health issues go.

Explore codebase

Worst Performer

The lowest code health score measured in the codebase. Points out long-term risks that you need to be aware of if the low performer is worked on.

View worst performers

Feedback icon

Spotbugs / Scope

Analysis Data

Project information, analysis scope, evolutionary data, and file content metrics.

Analysis Scope	Analysis Summary	Analysis Warnings	File Content	Overridden Code Health Rules	Commits	Issues
----------------	------------------	-------------------	--------------	------------------------------	---------	--------

Analysis Scope

Project	spotbugs
Includes History From	2003-03-24 15:43:47 (UTC)
Analyzed At	2024-11-20 18:40:07 (UTC)
Duration	8 minutes and 35 seconds

Spotbugs / Scope

Analysis Data

Project information, analysis scope, evolutionary data, and file content metrics.

Analysis Scope	Analysis Summary	Analysis Warnings	File Content	Overridden Code Health Rules	Commits	Issues
----------------	------------------	-------------------	--------------	------------------------------	---------	--------

Issues

Id	In Progress	FirstCommit	LastCommit	Done	CycleTime
BCEL-273		2016-06-08	2016-06-08		1h
BCEL-362		2024-01-13	2024-01-13		1h
BCEL-92		2016-06-08	2016-06-08		1h
BUG-605		2018-03-31	2018-03-31		1h
CVE-2021		2021-12-13	2021-12-13		1h
ISO-8859		2009-08-03	2009-08-03		1h
JDK-8169816		2017-07-18	2017-07-18		1h
JETTY-352		2007-06-04	2007-06-04		1h
JSR-14		2004-04-21	2004-04-21		1h
JSR-305		2007-08-17	2008-06-05		
PC-1		2024-01-20	2024-01-20		1h
UTF-8		2015-01-12	2023-10-16		

Spotbugs / Scope

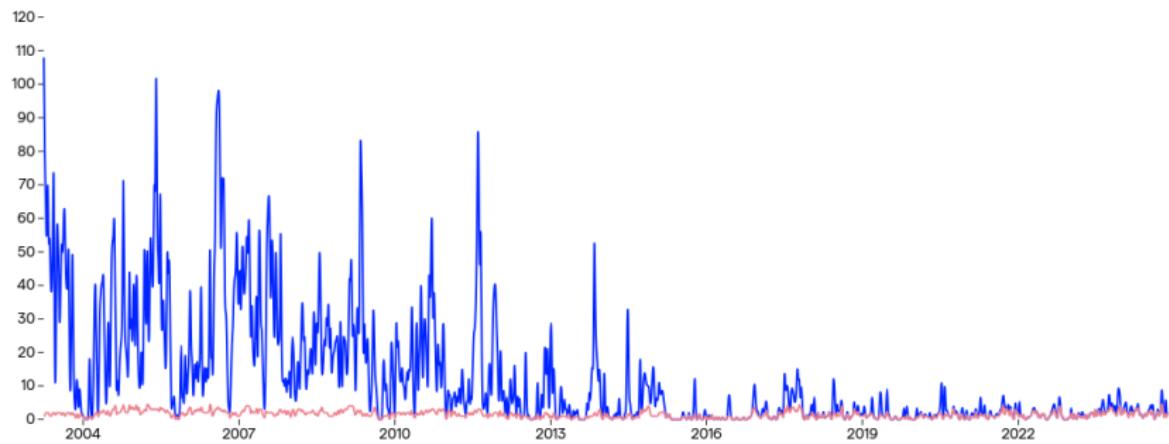
System Trends

[By Date](#)[By Task](#)[System Complexity](#)[Component Trends](#)[Code Age Trends](#)[Change Frequency Distribution](#)

Commit Activity Trend

— Number of Commits
— Number of Authors

Commits/Authors



Spotbugs / Scope

Code Churn by Task

The amount of changed code per task as defined by your Ticket IDs. [?](#)

By Date	By Task	System Complexity	Component Trends	Code Age Trends	Change Frequency Distribution		
This table shows the amount of changed code per task as defined by your Ticket IDs.							
	>Last Commit Date		Task ID	Added	Deleted	Net	Changed Files
2024-01-20			PC-1	200	4	196	4
2024-01-13			BCEL-362	55	4	51	8
2023-10-16			UTF-8	14	26	-12	6
2021-12-13			CVE-2021	1	1	0	1
2018-03-31			BUG-605	2	0	2	1
2017-07-18			JDK-8169816	1	1	0	1
2016-06-09			BCEL-273	11	13	-2	5
2016-06-08			BCEL-92	723	66	657	11
2009-08-03			ISO-8859	153	3	150	3
2009-01-13			SE-1	1	1	0	1
2008-06-05			JSR-305	750	74	676	44
2007-06-04			JETTY-352	89	0	89	2
2004-04-21			JSR-14	127	113	14	1
2004-03-26			JSR-166	11	3	8	2

Spotbugs / Hotspots

File-level Hotspots ?

Hotspots

Refactoring Targets

Costs

Defects

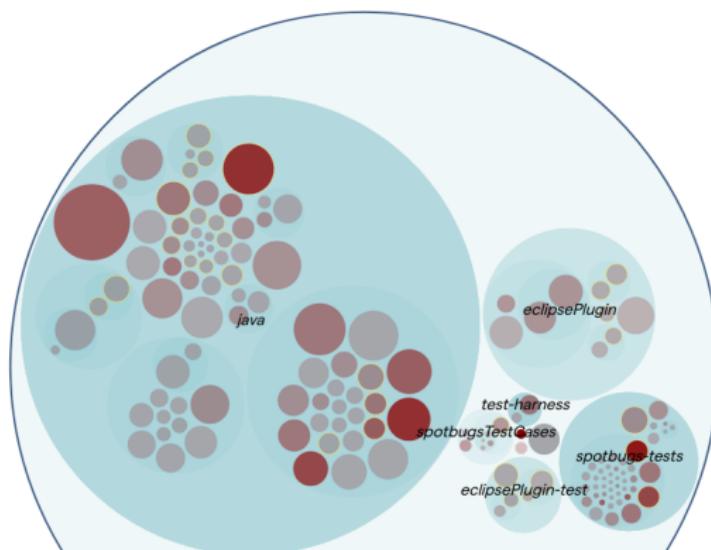
Programming Language

Code Health

System

ⓘ Low development activity

Hotspot



Search by filename

Filter by owners

Code Health Range

10.0

Commit Threshold ⓘ

2

Combined Aspects ⓘ

Hotspots Code Health Defects

System

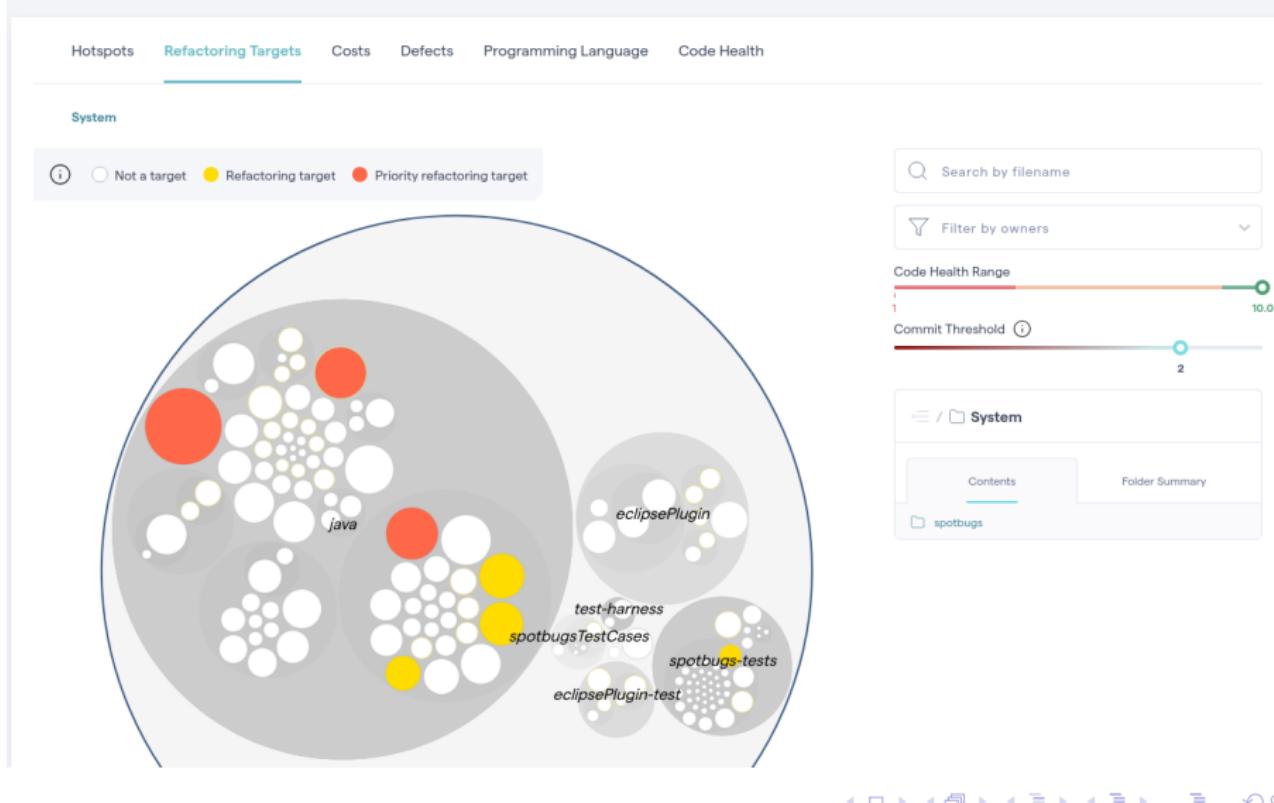
Contents

Folder Summary

spotbugs

Spotbugs / Hotspots

File-level Hotspots ?



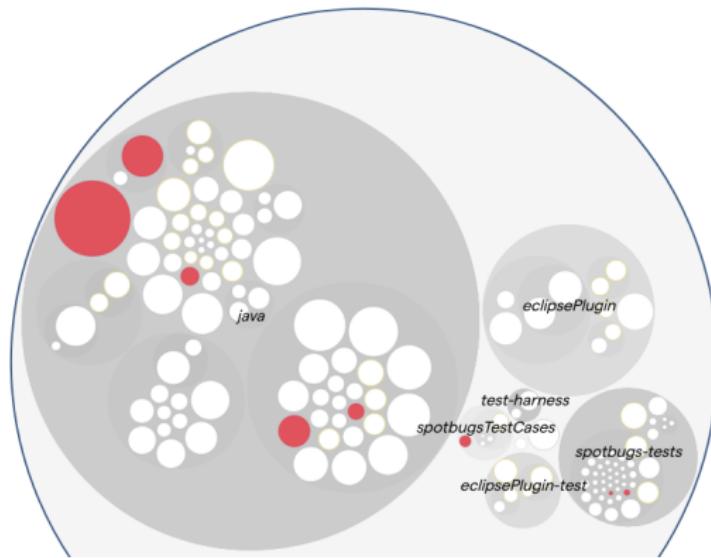
Spotbugs / Hotspots

File-level Hotspots ?

Hotspots Refactoring Targets Costs Defects Programming Language Code Health

System

ⓘ Low cost High cost



ⓘ Search by filename

ⓘ Filter by owners

Code Health Range

ⓘ 1 10.0

Commit Threshold ⓘ

ⓘ 2

ⓘ System

Contents Folder Summary

spotbugs



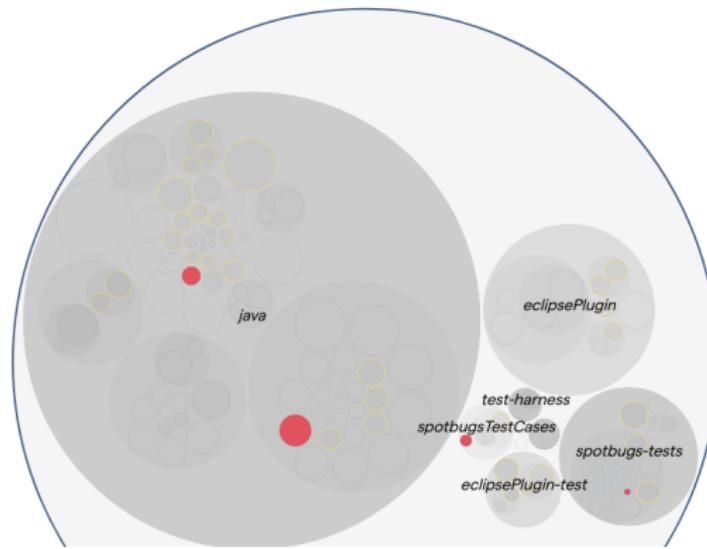
Spotbugs / Hotspots

File-level Hotspots ?

Hotspots Refactoring Targets Costs Defects Programming Language Code Health

System

ⓘ Fewer defects More defects



ⓘ Search by filename

ⓘ Filter by owners

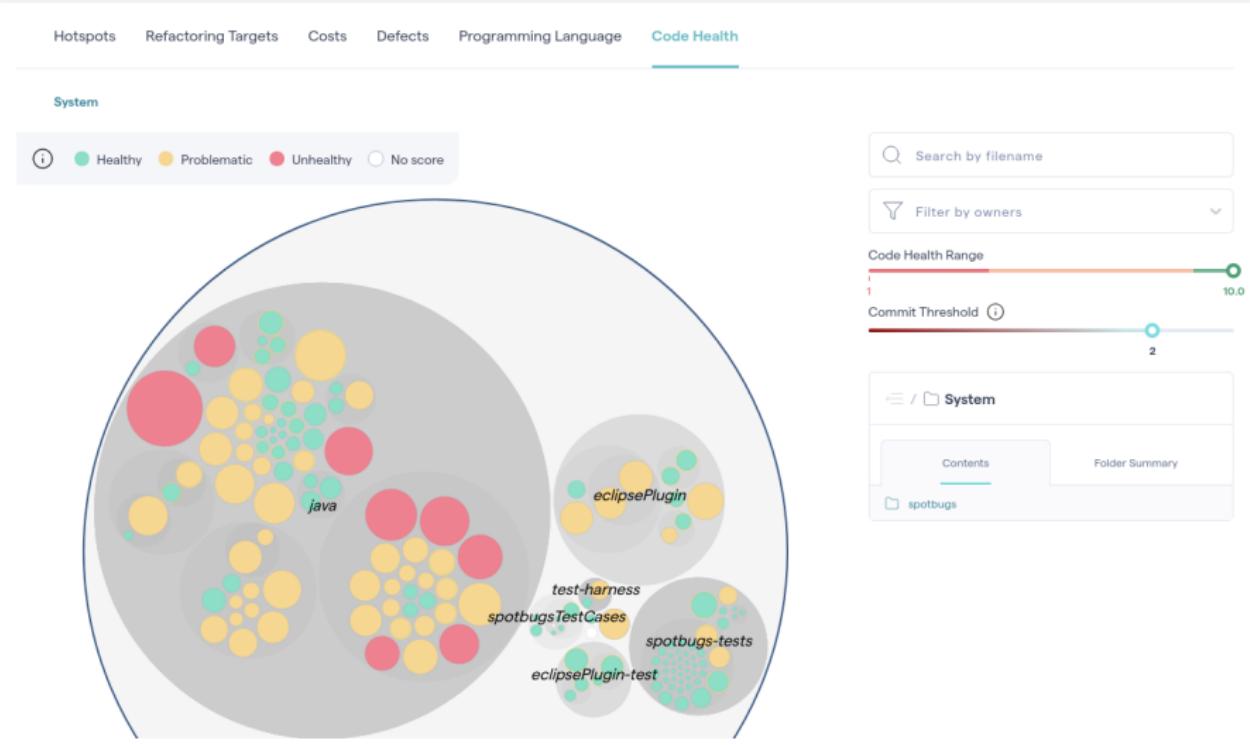
Code Health Range
1 10.0

Commit Threshold ⓘ 2

ⓘ System
Contents Folder Summary
 spotbugs

Spotbugs / Hotspots

File-level Hotspots ?



Spotbugs / Hotspots / Code Health

Hotspot Code Health

Track the health of key files in your codebase: hotspots and the files with goals. [?](#)

 Search file...[Filter by](#)[Code Health](#)

Hotspot

2023

Oct 2024

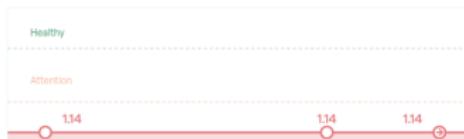
11/20/2024

Code Health Trend

Details

OpcodeStack.java

spotbugs/spotbugs/src/.../findbugs/
3100 LoC | 5 commits

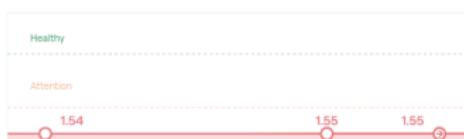
[Plan a goal](#)[X-ray](#)[Code Review](#)

- Potentially Low Cohesion
 - Modularity Issue
 - Bumpy Road
 - Deep, Nested Complexity
- [Read More](#)

[View Code](#)

UnreadFields.java

spotbugs/spotbugs/src/main/.../detect/
1097 LoC | 5 commits

[Plan a goal](#)[X-ray](#)[Code Review](#)

- Bumpy Road
 - Deep, Nested Complexity
 - Overall Code Complexity
 - Complex Method
- [Read More](#)

[View Code](#)

SerializableIdiom.java

spotbugs/spotbugs/src/main/.../detect/
663 LoC | 6 commits



- Bumpy Road
 - Deep, Nested Complexity
 - Many Conditionals
 - Complex Method
- [Read More](#)

Spotbugs / Hotspots / Code Health

File name: OpcodeStack.java • 1 File Code Health • Lines of Code: 3160

ⓘ What does the data in this table show?

Hotspots	Internal Change Coupling	Structural Recommendations	Change Frequency Distribution		
Function name	Change Frequency	Lines of Code	Code Review	Function Complexity	Actions
sawOpcode ⓘ	65	888	<ul style="list-style-type: none">Complex MethodBumpy Road AheadDeep, Nested ComplexityComplex Conditional	301	⋮
processMethodCall ⓘ	46	307	<ul style="list-style-type: none">Complex MethodBumpy Road AheadDeep, Nested ComplexityComplex Conditional	135	⋮
pushByIntMath ⓘ	15	151	<ul style="list-style-type: none">Complex MethodBumpy Road AheadComplex Conditional	73	⋮
OpcodeStack.Item.toString ⓘ	20	121	<ul style="list-style-type: none">Complex MethodBumpy Road Ahead	36	⋮
pushByLongMath	6	68	<ul style="list-style-type: none">Complex MethodBumpy Road AheadDeep, Nested ComplexityComplex Conditional	32	⋮

Spotbugs / Hotspots / Code Health

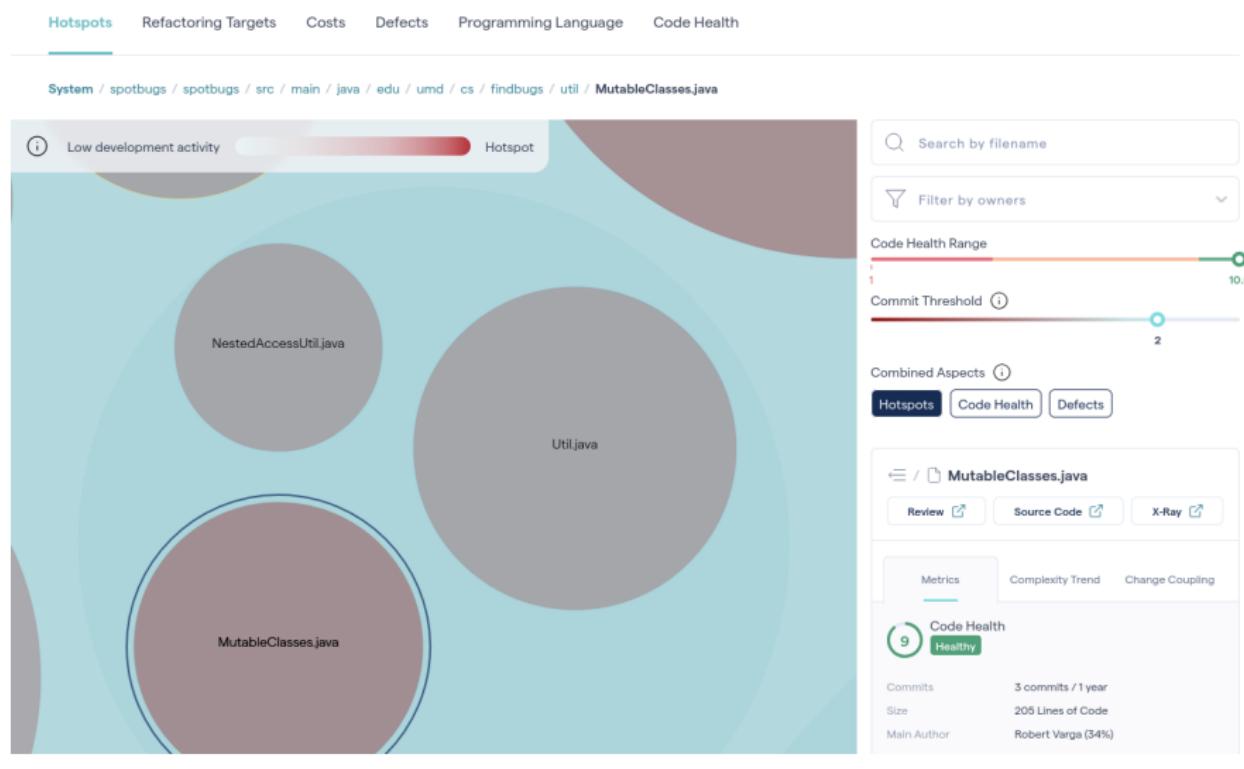
File name: OpcodeStack.java • 1 File Code Health • Lines of Code: 3160

ⓘ What does the data in this table show?

Hotspots	Internal Change Coupling	Structural Recommendations	Change Frequency Distribution
Coupled Functions	Degree of Coupling (%)	Average Revisions	Similarity (%)
<code>addJumpValue</code> <code>sawOpcode</code> <code>Compare</code>	27	40	0

Spotbugs / Hotspots / Code Health

File-level Hotspots ?



Spotbugs / Hotspots / Code Health

Hotspots / spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/util/MutableClasses.java

File name: MutableClasses.java • 9 File Code Health • Lines of Code: 205

What does the data in this table show?

Hotspots	Internal Change Coupling	Structural Recommendations	Change Frequency Distribution		
Function name	Change Frequency	Lines of Code	Code Review	Function Complexity	Actions
mutableSignature ⓘ	7	40	• Complex Method	10	⋮
MutableClasses.ClassAnalysis.computeByImmutableContract	1	19		6	⋮
MutableClasses.ClassAnalysis.getSig	1	7		2	⋮
MutableClasses.ClassAnalysis.looksLikeASetter	1	3		1	⋮
MutableClasses.ClassAnalysis.loadSuperAnalysis	0	20		5	⋮

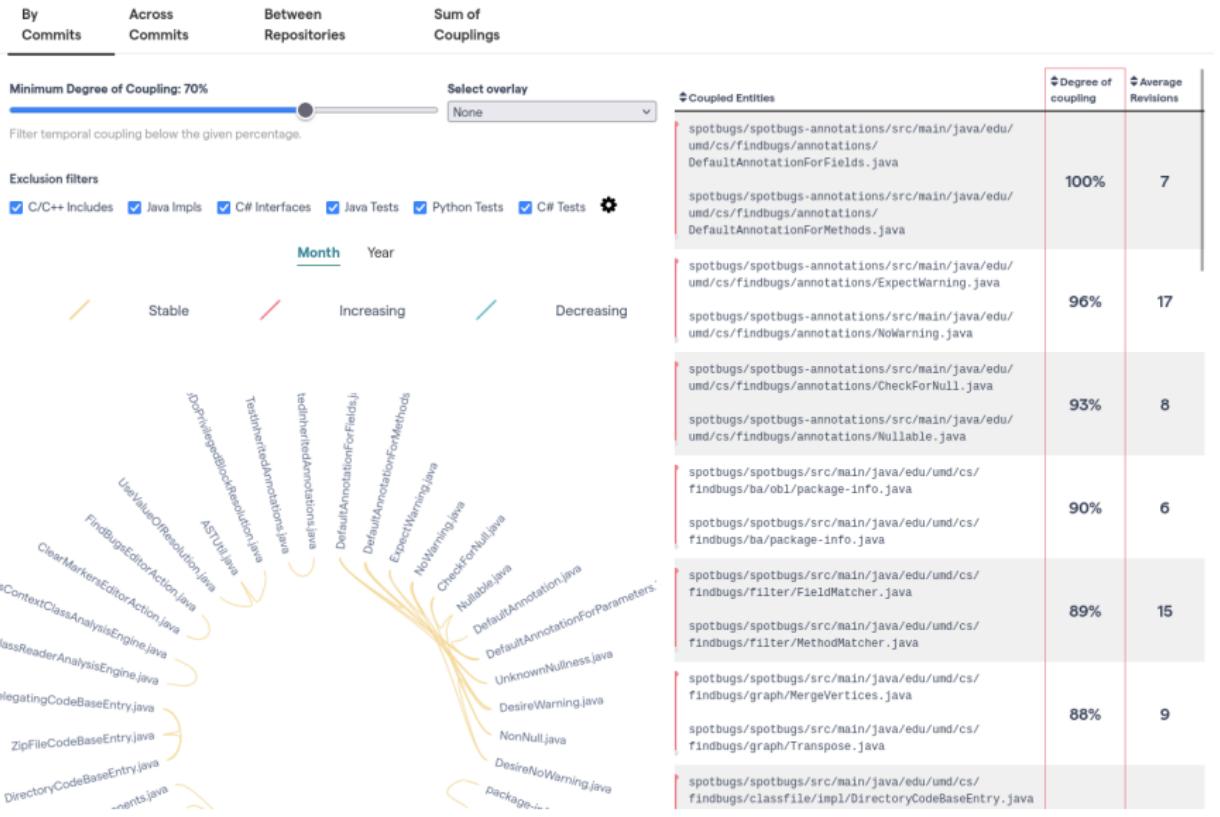
Spotbugs / Hotspots / Code Health

The screenshot shows the X-Ray IDE interface with the following details:

- Project:** mutableSignature
- Hotspots:** A list of detected issues:
 - Function Complexity: 10 (Complex)
 - Function Complexity: 6 (Medium)
 - Function Complexity: 2 (Simple)
 - Function Complexity: 1 (Very Simple)
 - Function Complexity: 5 (Medium)
 - Function Complexity: 6 (Medium)
- Code Preview:** The code for `MutableClasses.ClassAnalysis.computeMutable` is displayed, showing logic to determine if a class is mutable based on its name and annotations.

Function	Complexity	Actions
Function Complexity: 10	10	⋮
Function Complexity: 6	6	⋮
Function Complexity: 2	2	⋮
Function Complexity: 1	1	⋮
Function Complexity: 5	5	⋮
Function Complexity: 6	6	⋮

Spotbugs / Hotspots / Change coupling



Architectural analyses



Architectural components are not configured

With CodeScene's Architectural Components, you can:

- ✓ Provide CodeScene with your architectural knowledge

You know what your architecture should look like. CodeScene provides a rich suite of analyses to understand it both in terms of code and social patterns.

- ✓ Get a higher level of analysis

Architectural components provide a "big picture" view of a codebase. Which components are hotspots relative to other components and how is their Code Health?

- ✓ Track change coupling across architectural boundaries

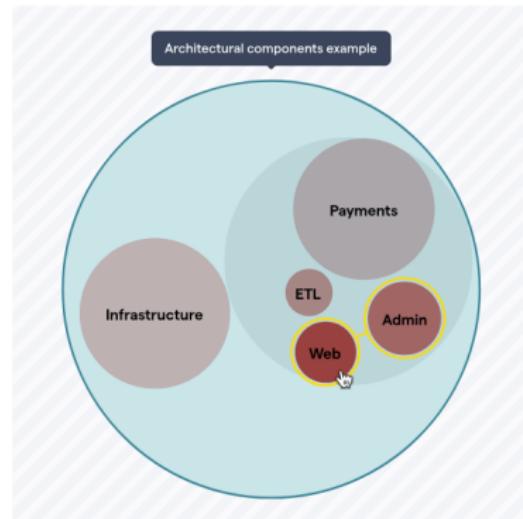
Which components change together?

- ✓ Get social insights

Do your components have clear team ownership? Which components have knowledge issues?

- ✓ Focus on individual components

Using System Health views, see which files are hotspots relative to their component. This can be a starting point for dealing with technical debt inside a subsystem.

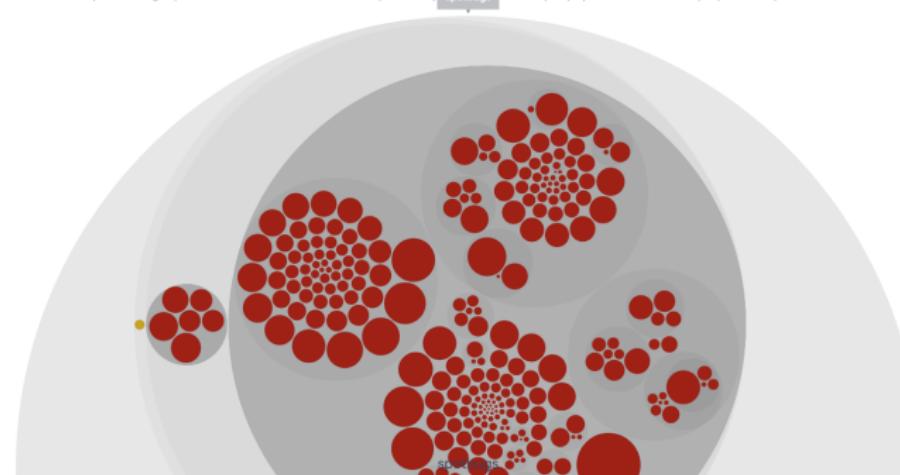


Spotbugs / Architecture / Recover

Architectural component editor

[Back to configuration](#)

Architectural components are groups of related files. How to define these components depends on the nature of your project and the kind of analysis you want to perform. This interface allows you to define them.



The screenshot shows the "Architectural Component Editor" interface. At the top right, there is a search bar labeled "Filter by filename..." and a commit frequency filter set to "1 commits". Below this, a section titled "Directory: System" is shown with the sub-instruction "Create an architectural pattern". A note says, "Use this directory's current path to add a new pattern to an existing architectural component, or to create a new component." There are two buttons: "Generate components" (which is highlighted in blue) and "Edit a custom pattern". Below these buttons is a text input field containing the character "/" and a "Save" button. At the bottom of the editor, there are three tabs: "Files", "Components", and "Colors". The "Components" tab is currently selected. At the very bottom of the page, there is a navigation bar with icons for back, forward, search, and refresh.

Architectural Component Editor

Filter by filename...

Commit frequency filter : 1 commits

Directory: System

Create an architectural pattern

Use this directory's current path to add a new pattern to an existing architectural component, or to create a new component.

Generate components

Edit a custom pattern

/

Files Components Colors

Directory: System

Spotbugs / Architecture / Recover

Search by name... Search by pattern... Not committed? Sort by: Name ▾

findbugs/asm

spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/asm/** ba Delete pattern

+ Add pattern manually Delete Component

findbugs/ba

spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/ba/** Delete pattern

+ Add pattern manually Delete Component

findbugs/bcel

spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/bcel/** Delete pattern

+ Add pattern manually Delete Component

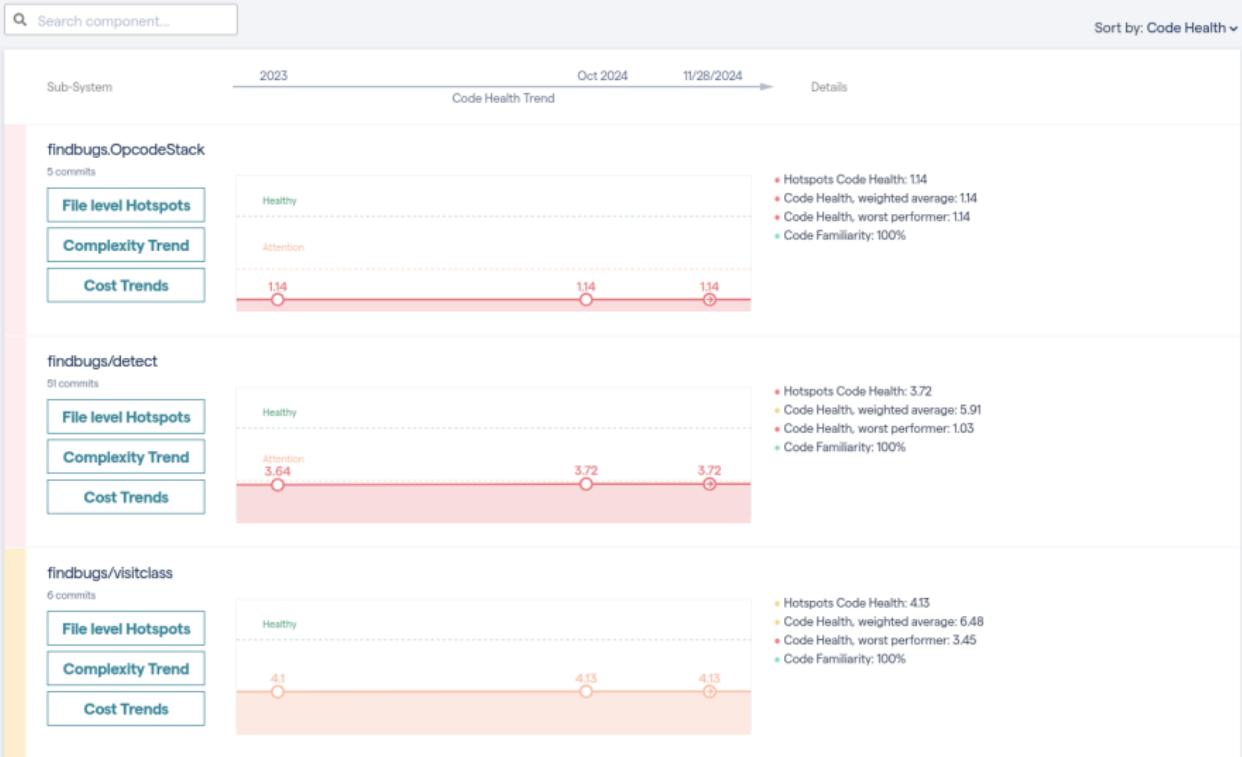
findbugs/bugReporter

spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/bugReporter/** Delete pattern

+ Add pattern manually Delete Component

Spotbugs / Architecture / Overview

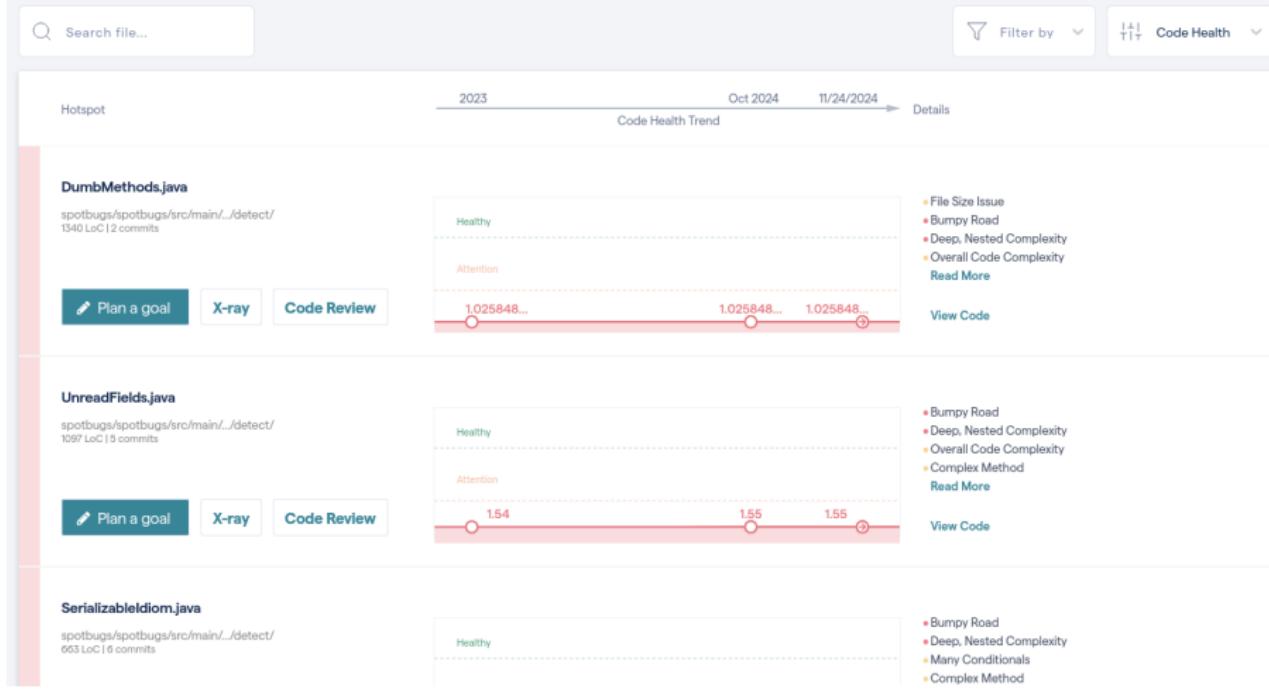
System Health



Spotbugs / Architecture / Details

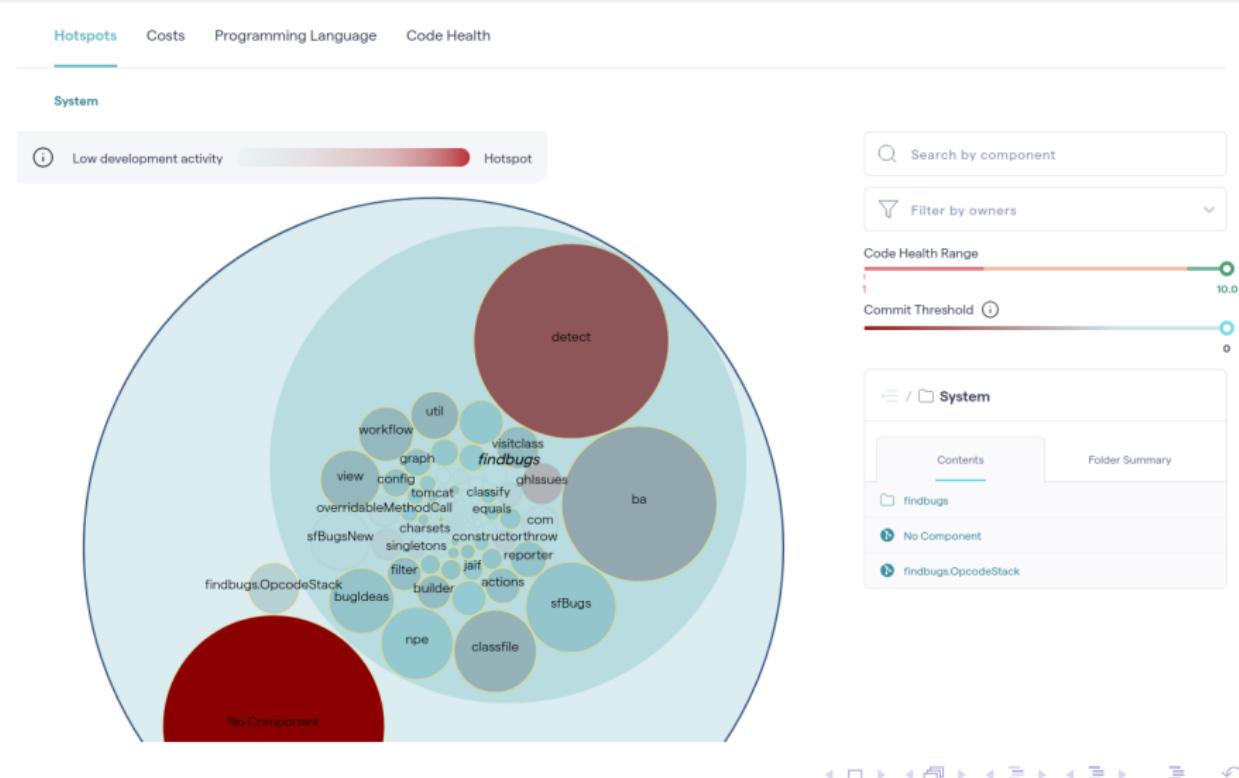
findbugs/detect: Architectural Component Hotspot Code Health

Code Health of your hotspots -- the most active parts of the codebase -- based on Code Biomarkers that detect properties of the code that indicate excess maintenance costs.



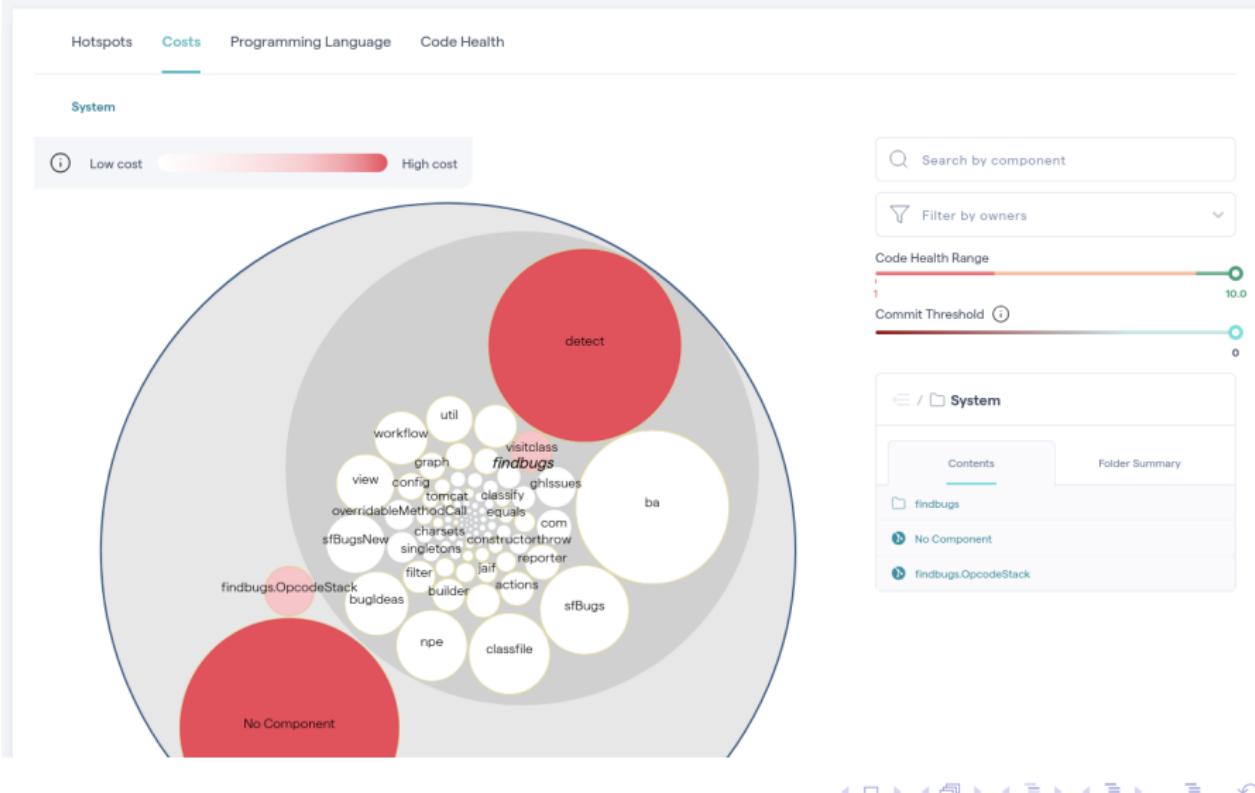
Spotbugs / Architecture / Hotspots

Architectural hotspots



Spotbugs / Architecture / Costs

Architectural hotspots



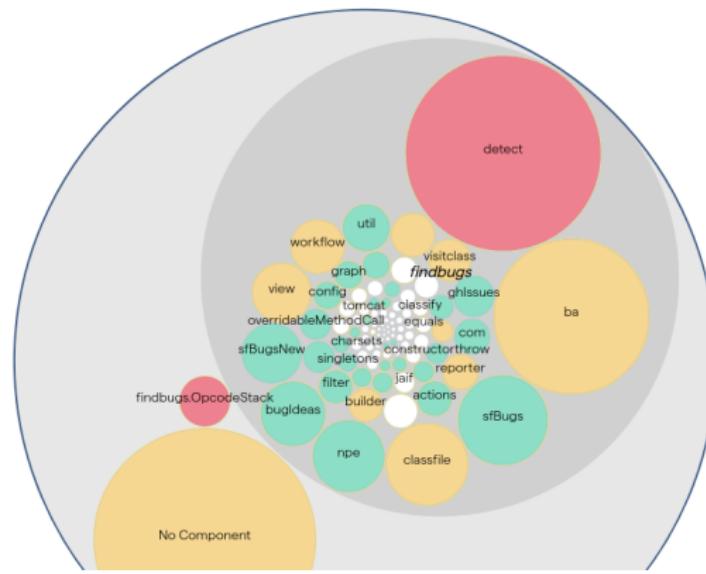
Spotbugs / Architecture / Code Health

Architectural hotspots

Hotspots Costs Programming Language Code Health

System

  Healthy  Problematic  Unhealthy  No score



 Search by component

 Filter by owners

Code Health Range
10.0

Commit Threshold  0

 /  System

Contents	Folder Summary
 findbugs	
 No Component	
 findbugs.OpcodeStack	

Brewday / Teams

Team-Code Alignment Explorer

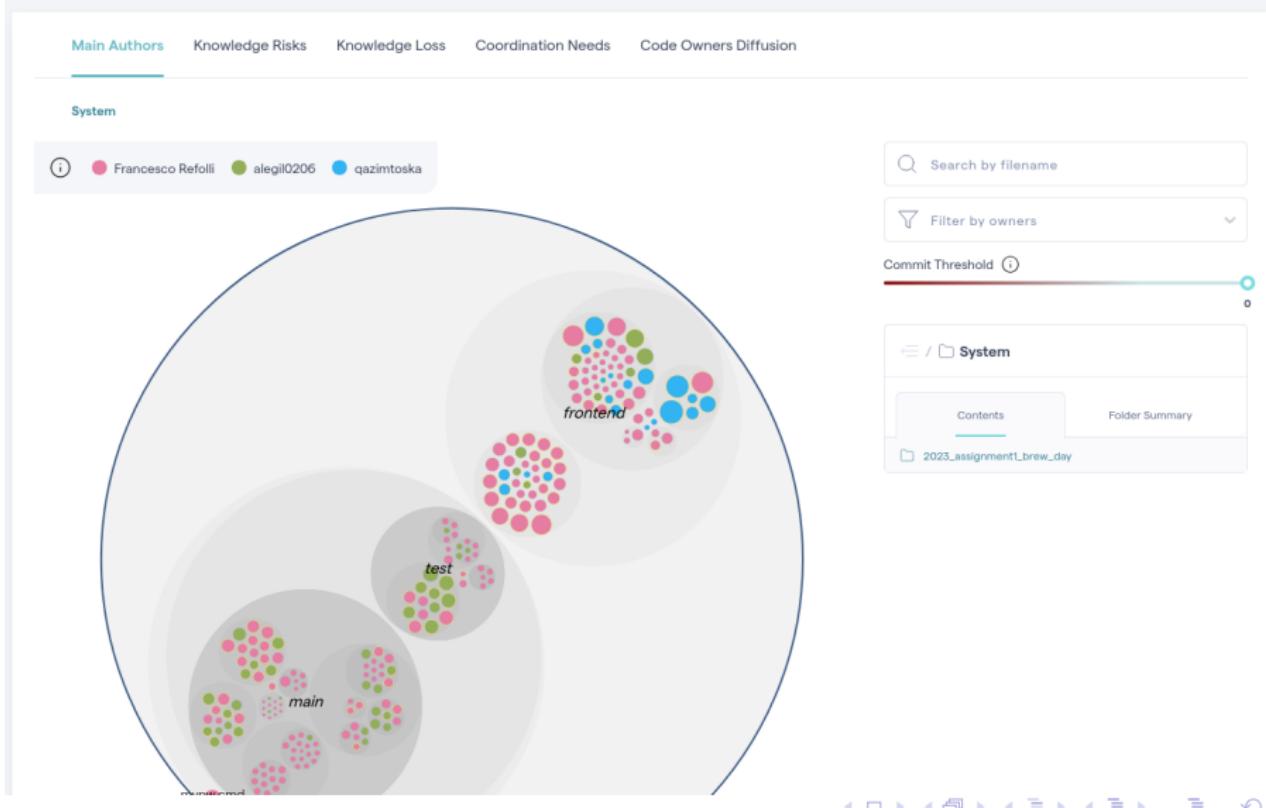
Explore how well aligned the development teams are with the software architecture. An efficient team structure minimizes the dependencies between teams while maintaining high team cohesion. In a cohesive team, members work on the same parts of the system.

ⓘ What does this graph show?



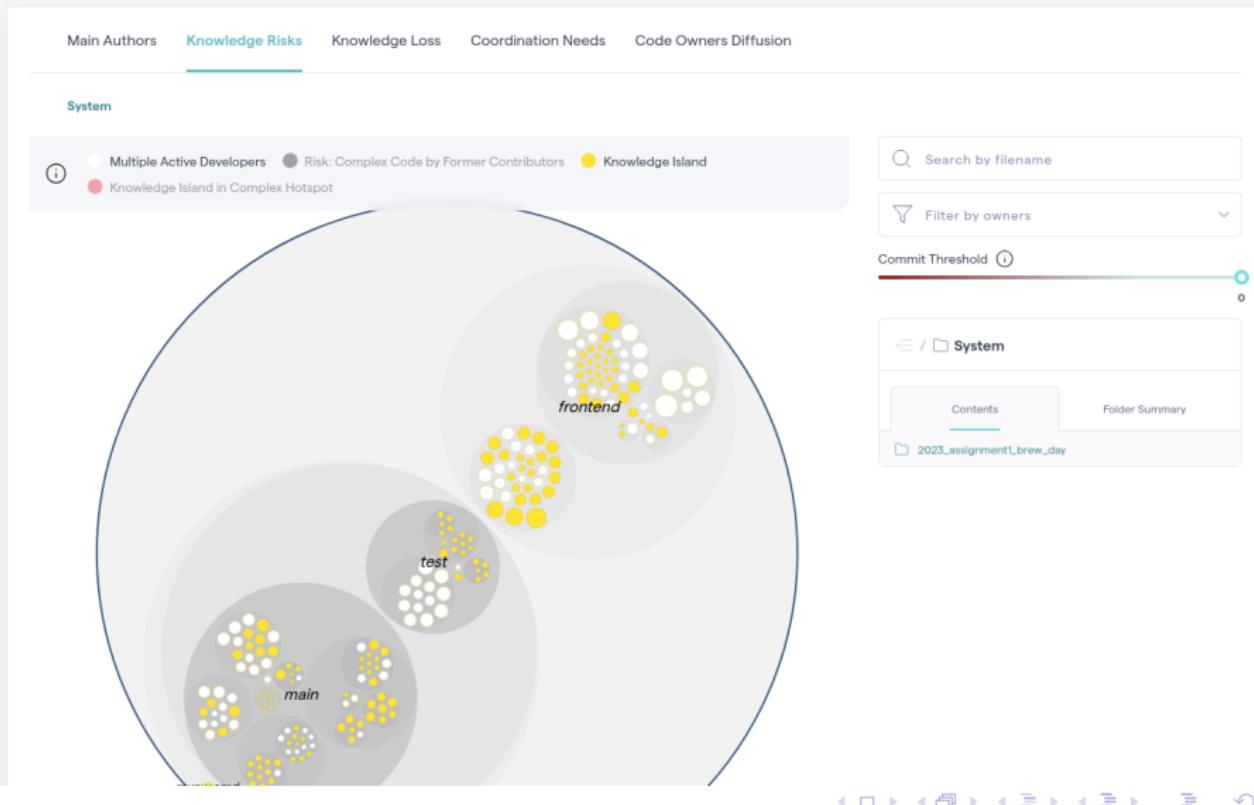
Brewday / Teams

Individuals ?



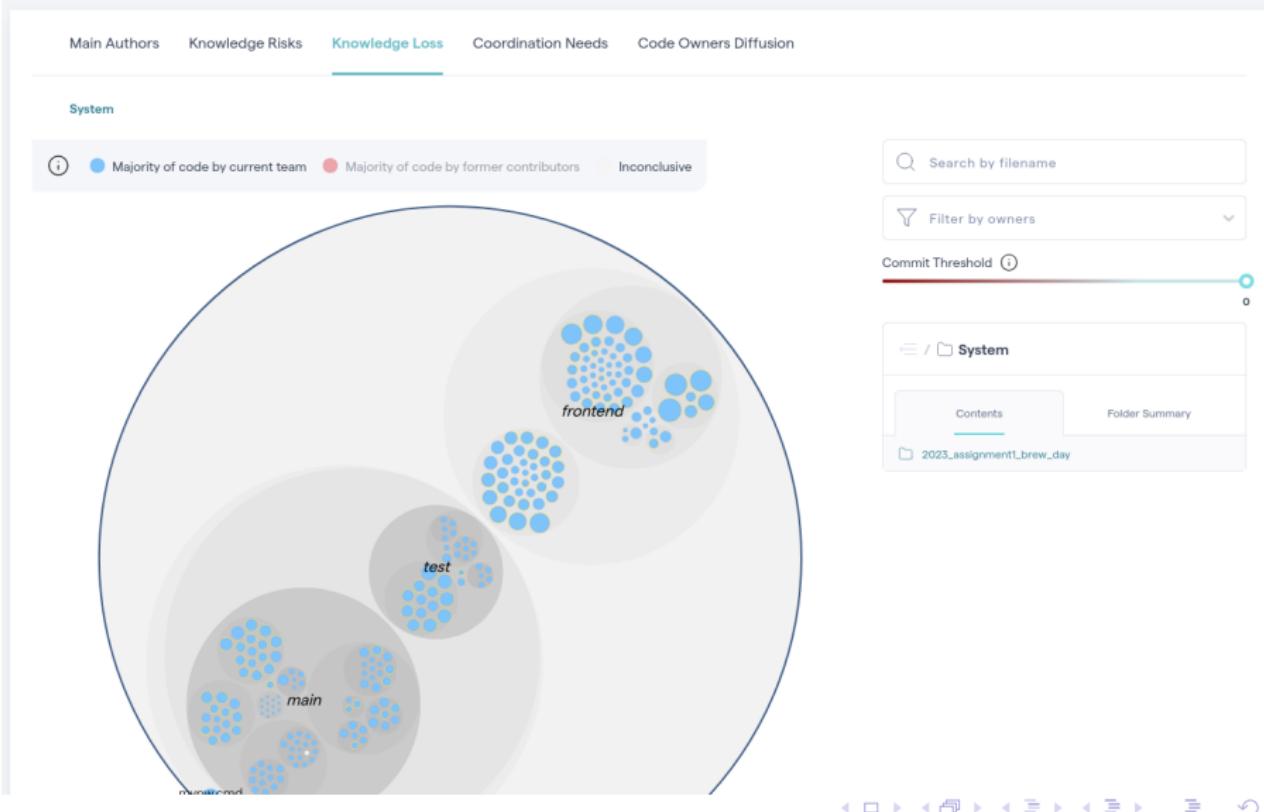
Brewday / Teams

Individuals ?



Brewday / Teams

Individuals ?



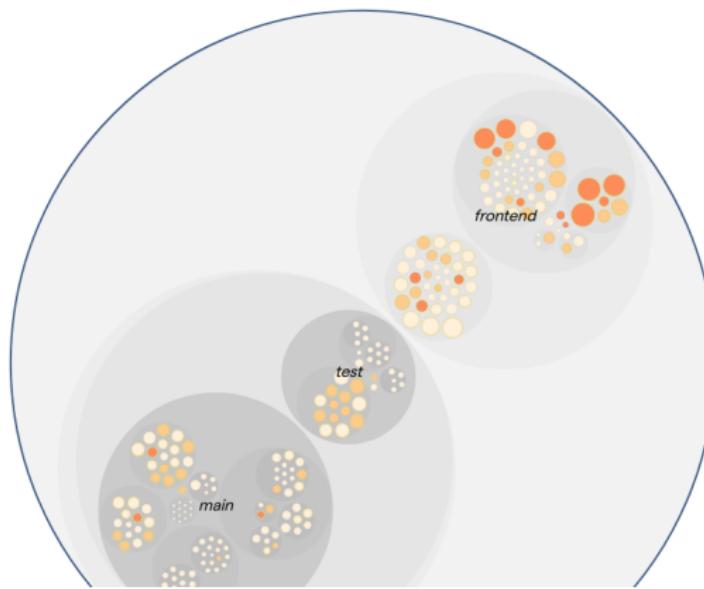
Brewday / Teams

Individuals ?

Main Authors Knowledge Risks Knowledge Loss **Coordination Needs** Code Owners Diffusion

System

(i) None Low Medium High



Search by filename

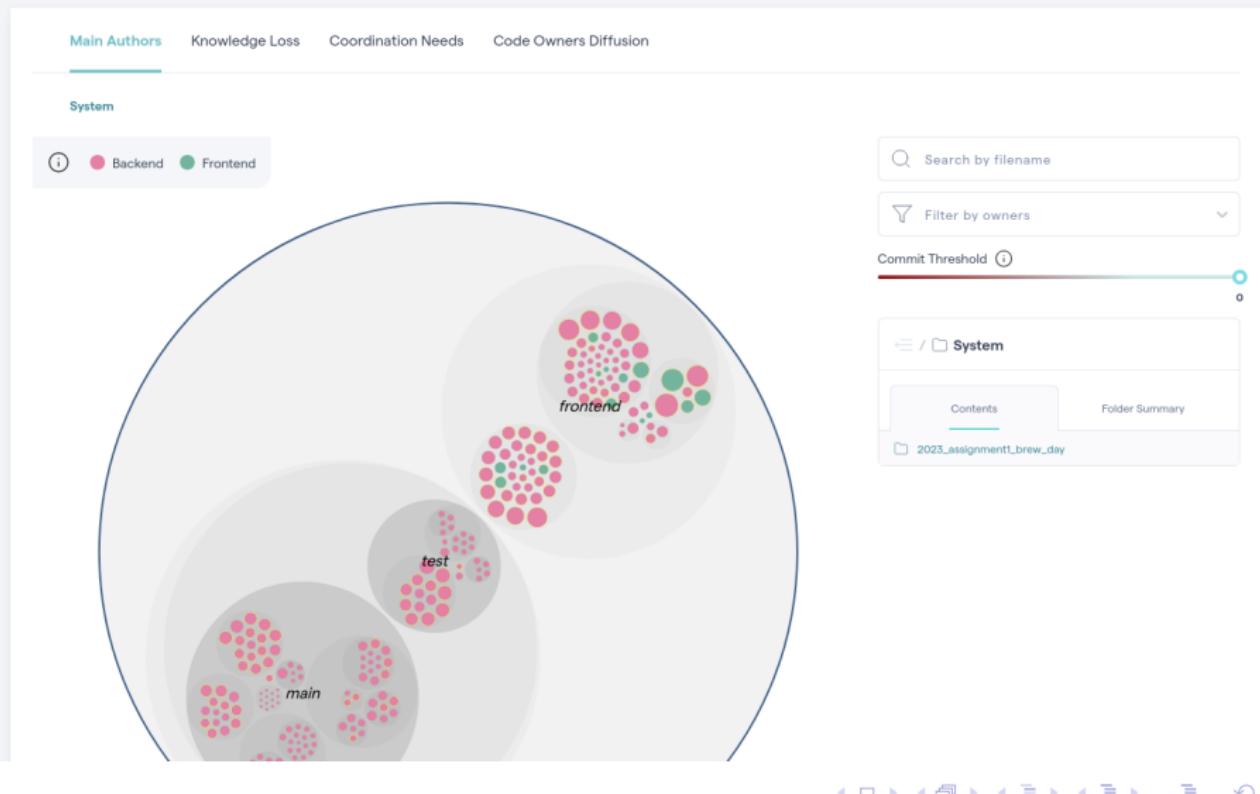
Filter by owners

Commit Threshold i

System
Contents i Folder Summary
2023_assignment1_brew_day

Brewday / Teams

Teams ?



Brewday / Teams

Author Statistics

A summary view of the contribution statistics and patterns per author. [?](#)

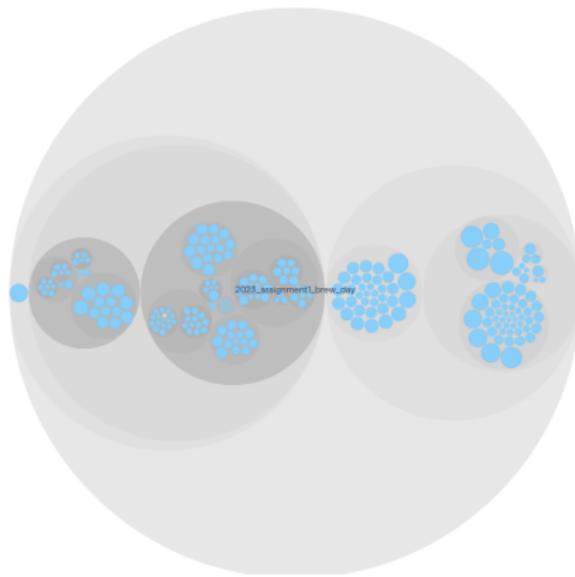
Duplicate authors can be merged in the [Author alias mapping interface](#).

Authors	Primary Component Owner	Primary File Owner	Commit Pattern	Former Contributor	Last Contribution	Months Contributing	Commits	LOC Added	LOC Removed	LOC Net
Francesco Refolfi	13	170	frequent		2023-02-20	0	172	11146	5117	6029
alegil0206	1	47	frequent		2023-02-20	0	175	4873	1019	3854
qazimtoska	1	21	frequent		2023-02-16	0	142	3042	1578	1464

Brewday / Simulations / Layoffs

Bus Factor: Off-boarding simulation

Simulate the code familiarity impact if developers leave. Is the system still maintainable? [?](#)



Legend

- Majority of code by current team
- Majority of code by former contributors
- Simulated off-boarding impact
- Off-Boarding Risk: hotspot with low code health
- Inconclusive (no data)

Filter by filename...

Commit frequency filter : 0 commits



Directory: System
 2023_assignment1_brew_day

Developers

Select one or more developers to simulate their departure from your organization.

Filter developer names:

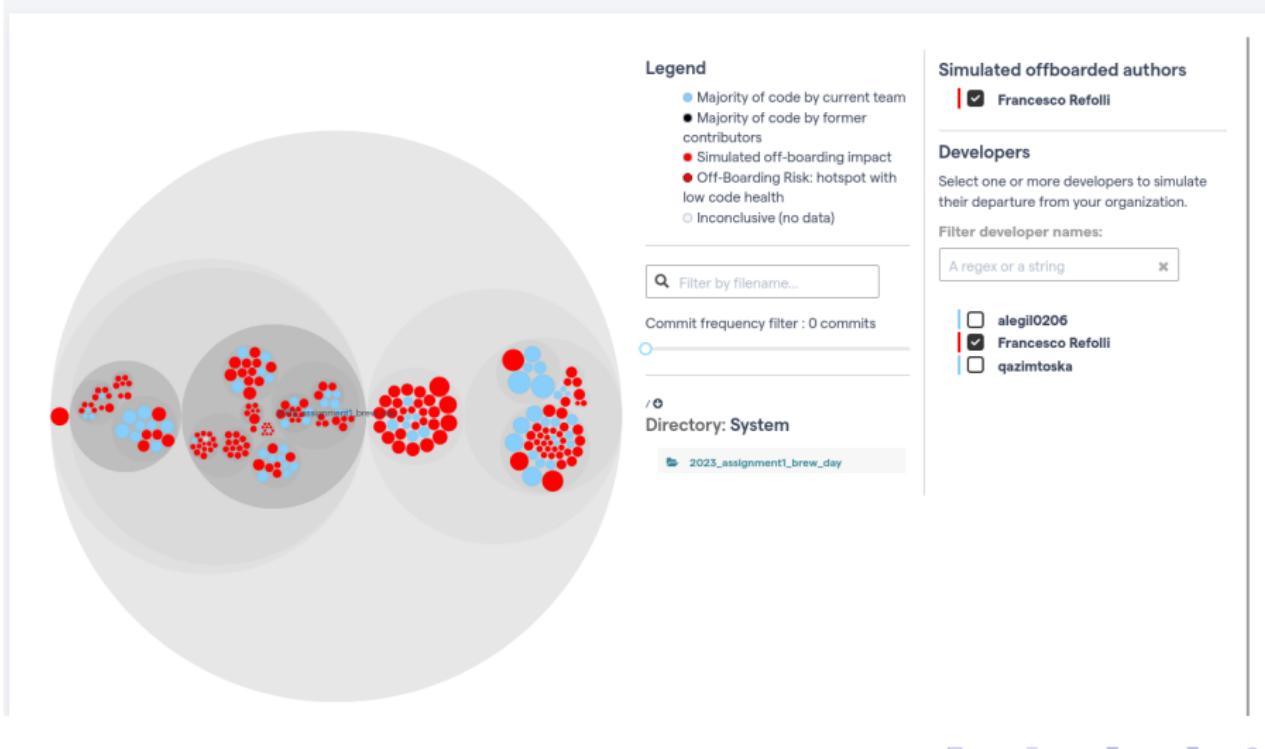
A regex or a string X

- alegil0206
- Francesco Refolli
- qazimtoska

Brewday / Simulations / Layoffs

Bus Factor: Off-boarding simulation

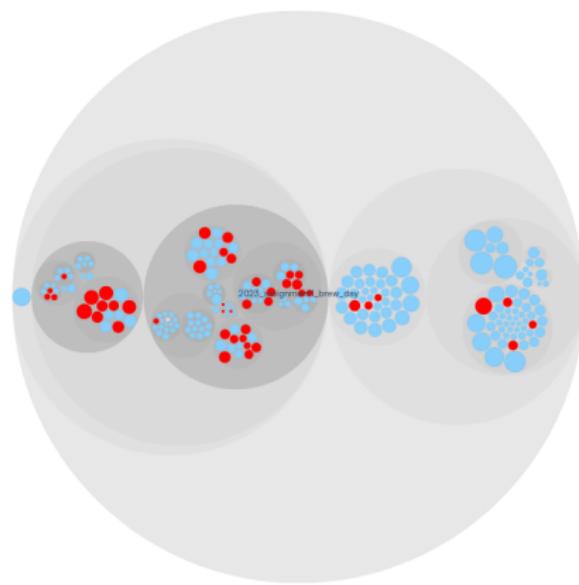
Simulate the code familiarity impact if developers leave. Is the system still maintainable? [?](#)



Brewday / Simulations / Layoffs

Bus Factor: Off-boarding simulation

Simulate the code familiarity impact if developers leave. Is the system still maintainable? [?](#)



Legend

- Majority of code by current team
- Majority of code by former contributors
- Simulated off-boarding impact
- Off-Boarding Risk: hotspot with low code health
- Inconclusive (no data)

Filter by filename...

Commit frequency filter : 0 commits



Directory: System

2023_assignment1_brew_day

Simulated offboarded authors

alegil0206

Developers

Select one or more developers to simulate their departure from your organization.

Filter developer names:

A regex or a string ✖

alegil0206

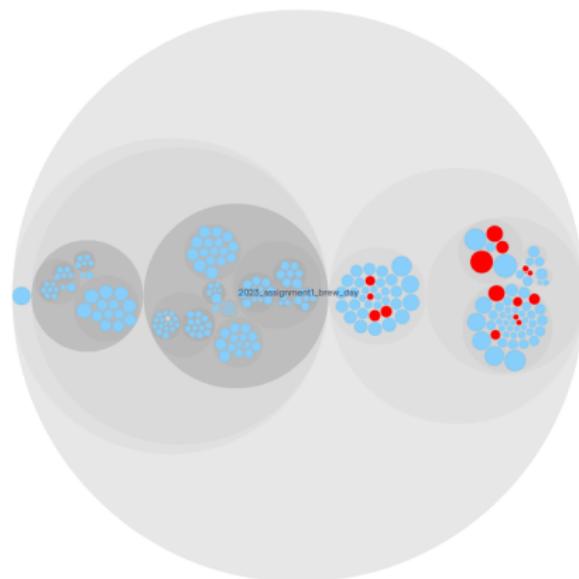
Francesco Refolfi

qazimtoska

Brewday / Simulations / Layoffs

Bus Factor: Off-boarding simulation

Simulate the code familiarity impact if developers leave. Is the system still maintainable? [?](#)



Legend

- Majority of code by current team
- Majority of code by former contributors
- Simulated off-boarding impact
- Off-Boarding Risk: hotspot with low code health
- Inconclusive (no data)

Filter by filename...

Commit frequency filter : 0 commits



Directory: System
2023_assignment1_brew_day

Simulated offboarded authors

qazimtoska

Developers

Select one or more developers to simulate their departure from your organization.

Filter developer names:

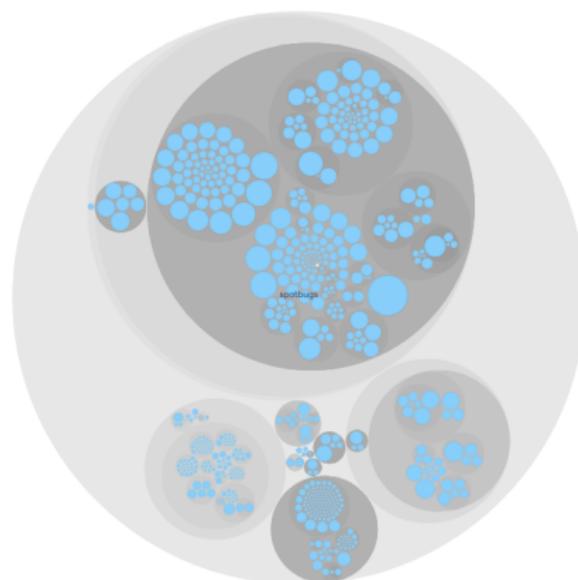
A regex or a string

alegilO206
 Francesco Refolli
 qazimtoska

Spotbugs / Simulations / Layoffs

Bus Factor: Off-boarding simulation

Simulate the code familiarity impact if developers leave. Is the system still maintainable? [?](#)



Legend

- Majority of code by current team
- Majority of code by former contributors
- Simulated off-boarding impact
- Off-Boarding Risk: hotspot with low code health
- Inconclusive (no data)

Filter by filename...

Commit frequency filter : 1 commits



/o
Directory: System

spotbugs

Developers

Select one or more developers to simulate their departure from your organization.

Filter developer names:

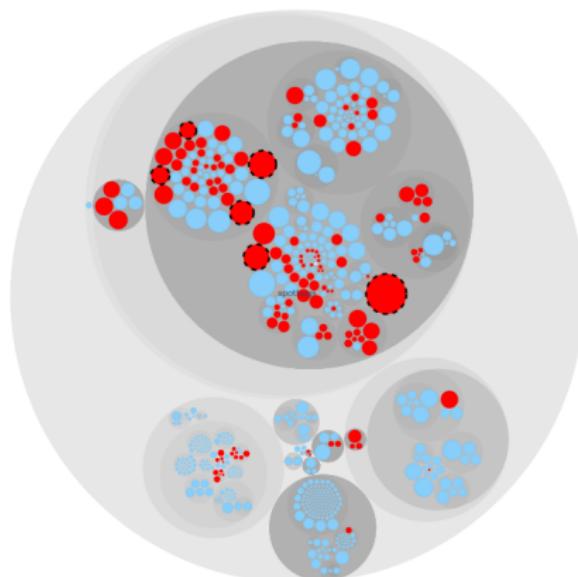
A regex or a string X

- Adrian Turtozcki
- Akira Ajiaska
- Alex Mont
- Alex-Vol-Amz
- andi
- Andi Pabst
- Andreas Hubold
- Andreas Sewe
- Andrey Loskutov
- Andy Coates
- Anemone95
- Balogh, Ádám
- Benjamin Langmead
- Bill
- Bill Pugh
- Brian Cole
- Brian Goetz
- Brian Riehman
- Carsten Pfeiffer

Spotbugs / Simulations / Layoffs

Bus Factor: Off-boarding simulation

Simulate the code familiarity impact if developers leave. Is the system still maintainable? [?](#)



Legend

- Majority of code by current team
- Majority of code by former contributors
- Simulated off-boarding impact
- Off-Boarding Risk: hotspot with low code health
- Inconclusive (no data)

Filter by filename...

Commit frequency filter : 1 commits



/o
Directory: System

spotbugs

Simulated offboarded authors

Bill Pugh

Developers

Select one or more developers to simulate their departure from your organization.

Filter developer names:

A regex or a string ✖

- Adrian Turtockzki
- Akira Aljsaka
- Alex Mont
- Alex-Vol-Amz
- andi
- Andi Pabst
- Andreas Hubold
- Andreas Sewe
- Andrey Loskutov
- Andy Coates
- Anemone95
- Balogh, Ádám
- Benjamin Langmead
- Bill
- Bill Pugh
- Brian Cole
- Brian Goetz
- Brian Riehman

Projects Report

spotbugs

Last analysis: Nov 28, 2024, 10:59:00 AM

Code Health ⓘ Attention ↗ Improving

Knowledge Distribution ⓘ Healthy No change

Team-Code AI ⓘ N/A No change

[Run Analysis](#)

CONFIGURATION

- Project configuration
- Pull request integration
- Project management integration

[Dashboard →](#)

lartc

Last analysis: Nov 28, 2024, 12:44:42 PM

Code Health

MONTHLY TRENDS

Code Health ⓘ Attention ↗ Improving

Knowledge Distribution ⓘ Attention No change

N/A No change N/A No change

[Generate PDF report](#)

REPORTS

[Dashboard →](#)

Navigation icons: back, forward, search, etc.

Projects Report

[← BACK TO PROJECTS](#)

PDF Reports

Generate a report now or schedule to receive reports once a day, week, month or year.

 Project Level: spotbugs



Code Health Overview Report

Get an overview of Code Health metrics, trends and alerts for a specific project.

[Generate Report](#)



 All Projects



Software Portfolio Overview Report

Get visibility into your entire portfolio. This high level portfolio overview highlights the status of each project in terms of Code Health, Knowledge Distribution, Team-Code Alignment and Delivery.

[Generate Report](#)



Summary #1

Is the tool useful in terms of understanding the code, design, or /and architecture? Does it allow you to reconstruct the software architecture of a system?

- ① "Usefulness" is always a matter of case to case evaluation. It certainly highlights worst files or functions in the code base, but that is counterbalanced with interface limitations.
- ② Kind of, it enables a structural (as in directory-file) view over code, but definition of components is left to the user, so a certain degree of knowledge is needed.

Summary #2

Does it allow to identify anomalies at the code level (code smells)? At the level of design or architecture (design and architectural smell)? If so, which ones?

- ① Yes, it identifies code-level/file-level code smells (or sort of).
- ② Information leaked about architecture is only in the form of computation on top of files (these are components) and change coupling.

Summary #3

Does it allow you to identify the most critical problems and then prioritize them (ranking)? Or to define thresholds for critical metrics? Does it allow you to change the metric thresholds?

- ① Yes, it lists files or functions by Code Health (which is the metric).
- ② That said, other metrics are usually only filters (such as commit threshold).

Summary #4

Is the tool able to detect the duplicate code?

- ① No.

Summary #5

Is the tool able to detect cyclical dependency? At the package and class level?

- ① No, it doesn't elaborate dependencies.
- ② It could be inferred from change coupling but it's an edge case and it's left to the user.

Summary #6

Is the tool able to compute a technical debt index? Which one (s)?

- ① It computes Code Health as sum of some metrics about LoC and smells.
- ② It computes also the Cost of maintenance which is estimated with number of commits containing bugs or fixes (requires labelled data!).

Summary #7

Does the tool provide quality gates?

- ➊ No, but it can be used in CI (the site says so but I haven't tested it).

Summary #8

Does the tool allow you to evaluate the history of a project? Does it provide analysis on the history of a project? Trend analysis?

- ① Yes, the analysis is carried out on the whole commit history of the project, and sliding windows can be customized to analyze only a part of it or in chunks.

Summary #9

Does it provide support for refactoring issues? Refactoring actions

- ① No, it only serves an indicator of coupling/proximity for functions (which can be useful or not depending on the situation).

Summary #10

Are reports generated automatically? Is it useful for Continuous integration?

- ① Yes, it can generate two reports automatically, however they contain only summary data about a project or all projects.
- ② Again, usefulness isn't absolute, and i tend for a No.

Summary #11

Which are the main advantages and disadvantages? What is missing by the tool in your opinion?

- ① It highlights hotspots, it allows to prioritize files by code health and evaluate risk of knowledge islands.
- ② But it totally lacks construct semantics integration.
- ③ Architectural Component definition process is tedious and for most projects with regular structure it could be inferred automatically (with construct semantics or file structure).
- ④ I "can" see this product used along with SonarQube, definitely not replacing it at all.

The End (QA?)

