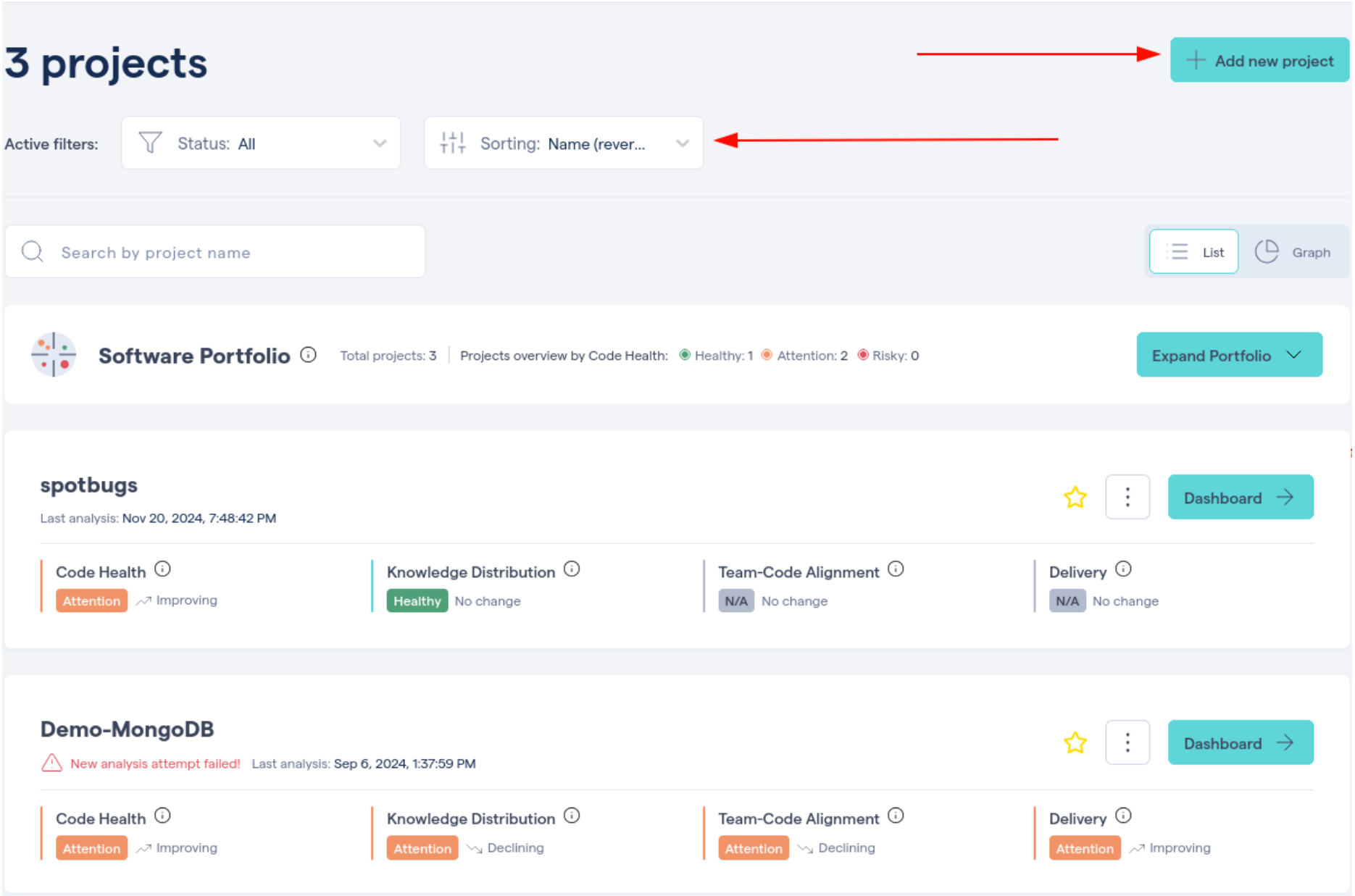
CodeScene.io

Software Evolution and Reverse Engineering

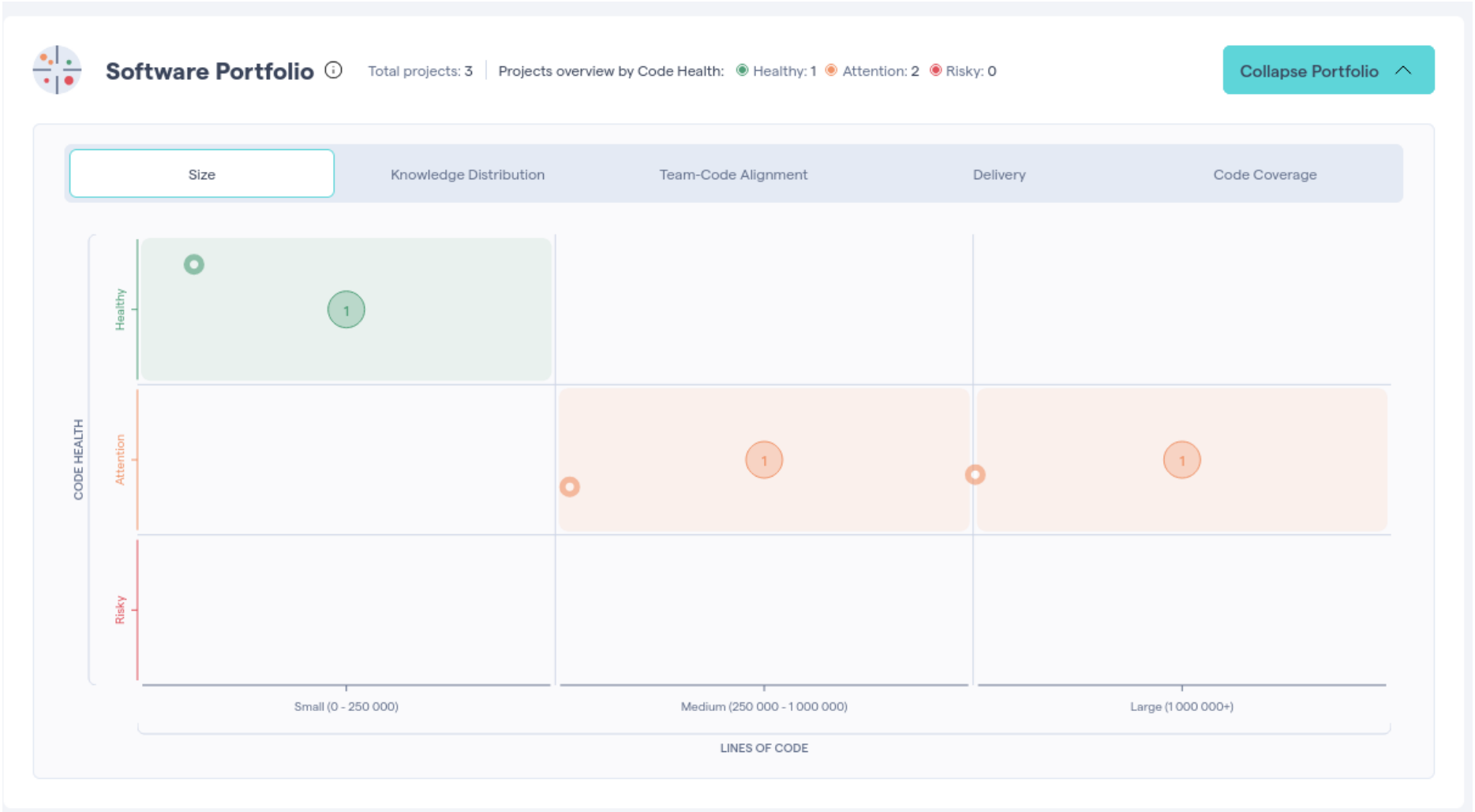
Refolli F. 865955

March 2, 2025

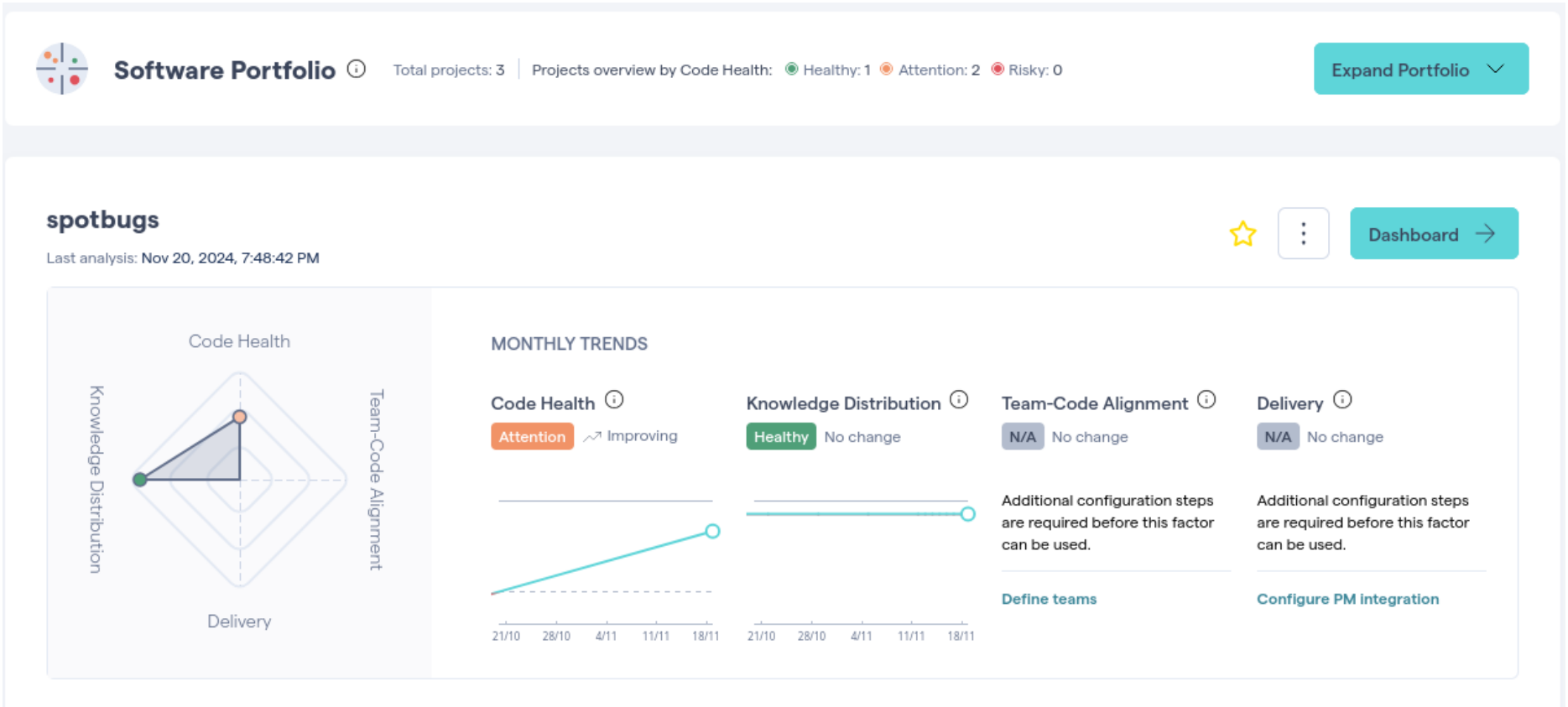
Projects Overview



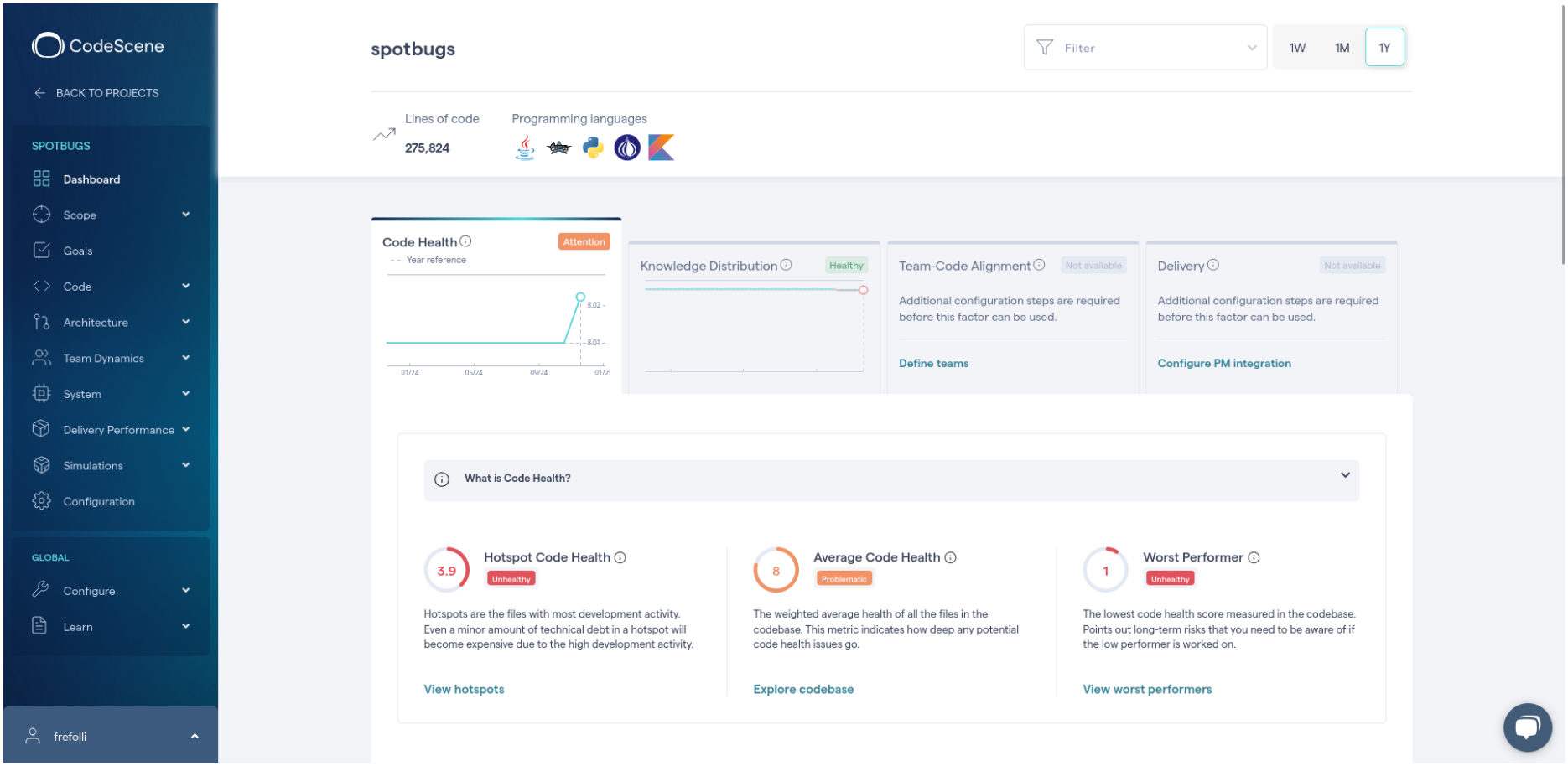
Projects Overview



Projects Overview



Spotbugs / Dashboard



Spotbugs / Scope

Analysis Data

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

Analysis Scope | Analysis Summary | Analysis Warnings | File Content | Overriden 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 | Overriden Code Health Rules | Commits | Issues

Issues

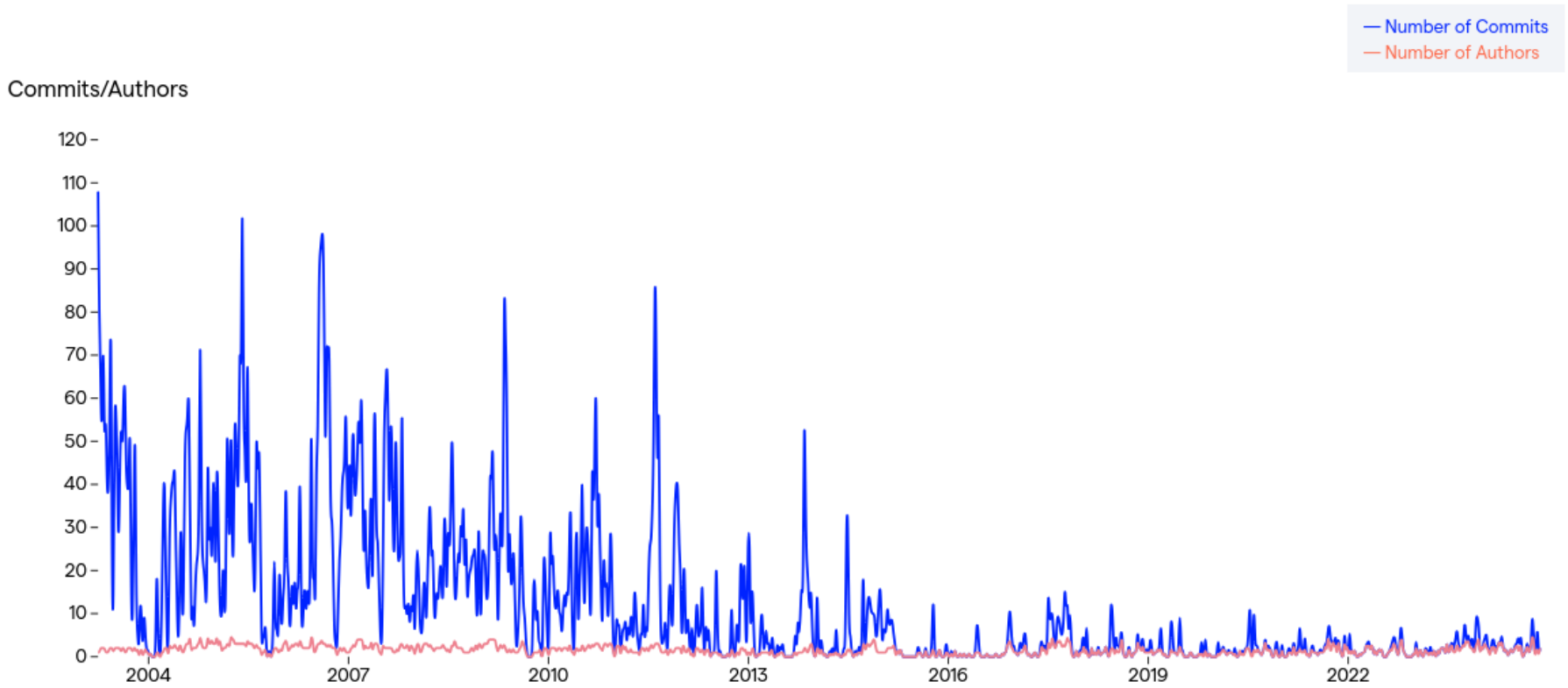
Id	In Progress	FirstCommit	LastCommit	Done	CycleTime
BCEL-273		2016-06-08	2016-06-08		1h
BCEL-362		2024-01-13	2024-01-13		1h
BCEL-92		2016-06-08	2016-06-08		1h
BUG-605		2018-03-31	2018-03-31		1h
CVE-2021		2021-12-13	2021-12-13		1h
ISO-8859		2009-08-03	2009-08-03		1h
JDK-8169816		2017-07-18	2017-07-18		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



Spotbugs / Scope

Code Churn by Task

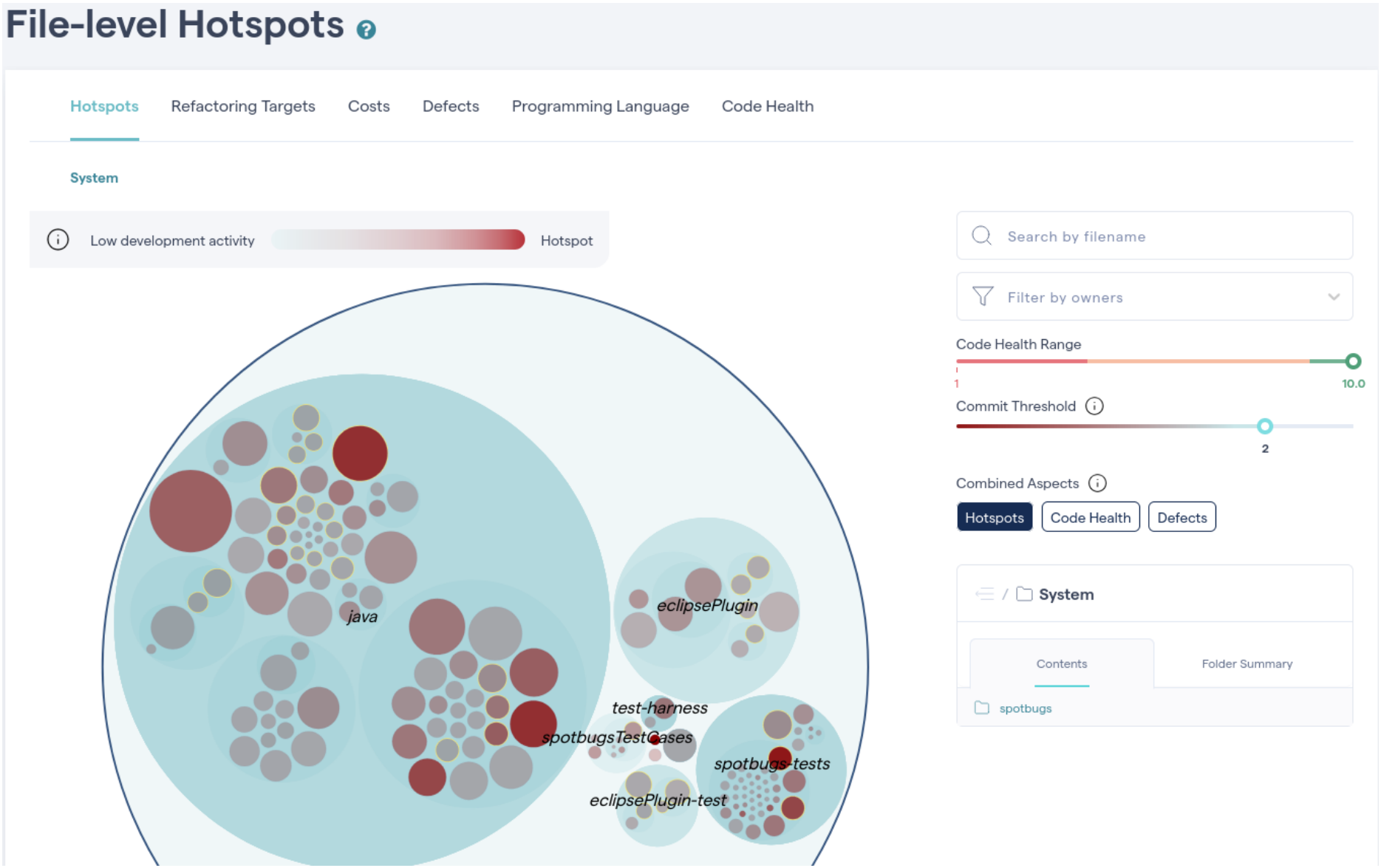
The amount of changed code per task as defined by your Ticket IDs.

By Date | By Task | System Complexity | Component Trends | Code Age Trends | Change Frequency Distribution

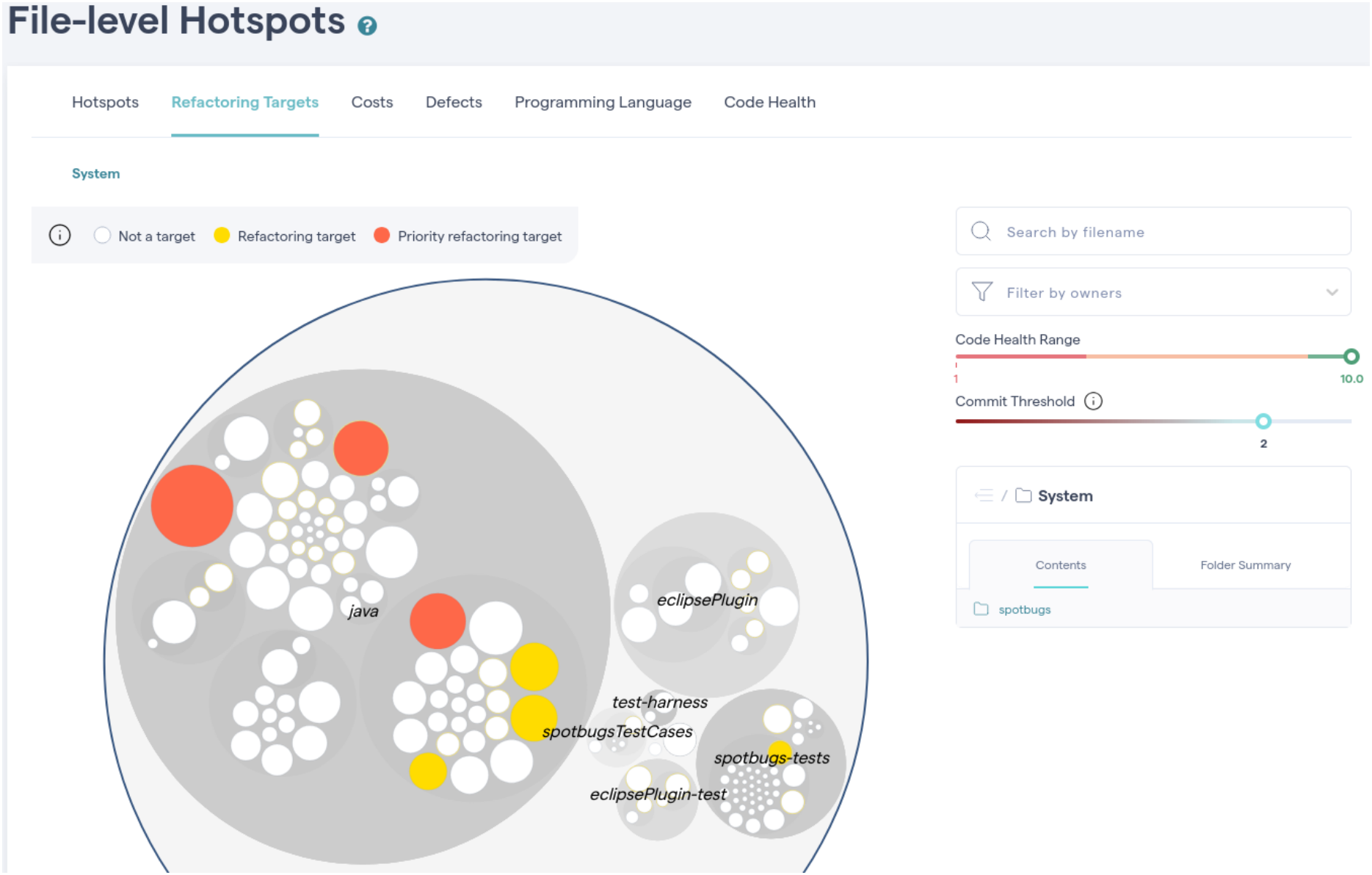
This table shows the amount of changed code per task as defined by your Ticket IDs.

Last Commit Date	Task ID	Added	Deleted	Net	Changed Files	Authors
2024-01-20	PC-1	200	4	196	4	1
2024-01-13	BCEL-362	55	4	51	8	1
2023-10-16	UTF-8	14	26	-12	6	4
2021-12-13	CVE-2021	1	1	0	1	1
2018-03-31	BUG-605	2	0	2	1	1
2017-07-18	JDK-8169816	1	1	0	1	1
2016-06-09	BCEL-273	11	13	-2	5	1
2016-06-08	BCEL-92	723	66	657	11	1
2009-08-03	ISO-8859	153	3	150	3	1
2009-01-13	SE-1	1	1	0	1	1
2008-06-05	JSR-305	750	74	676	44	2
2007-06-04	JETTY-352	89	0	89	2	1
2004-04-21	JSR-14	127	113	14	1	1
2004-03-26	JSR-166	11	3	8	2	1

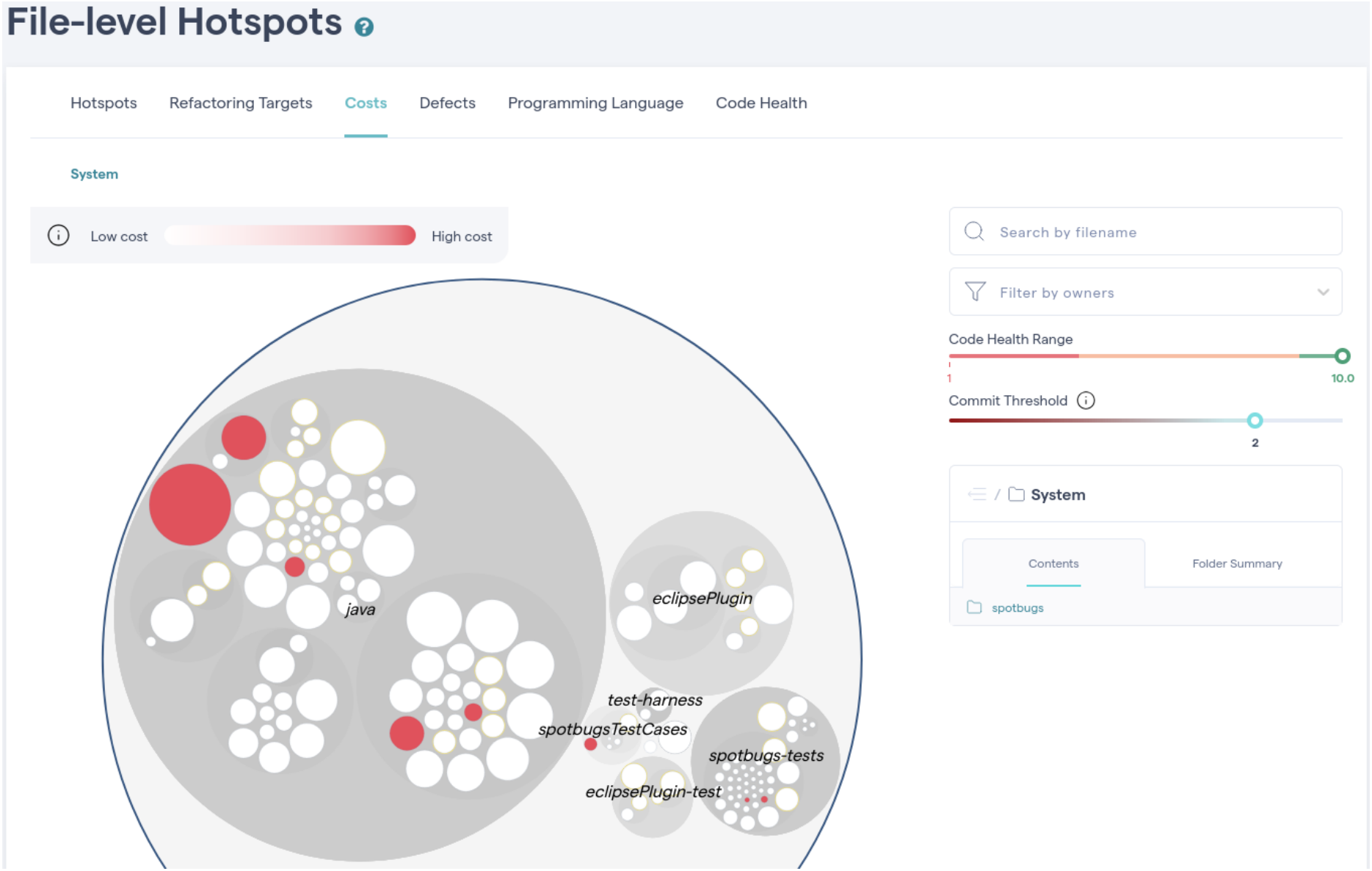
Spotbugs / Hotspots



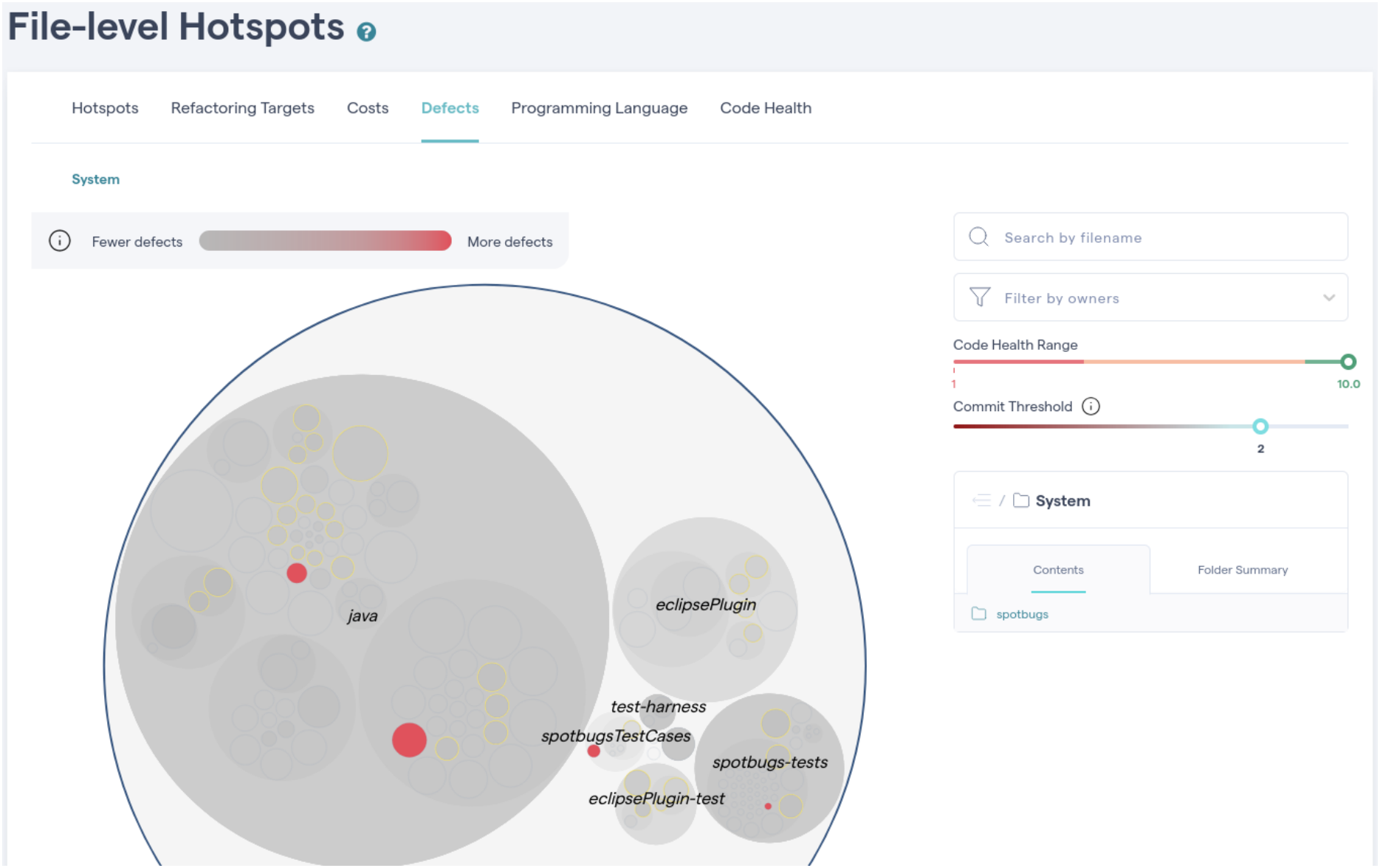
Spotbugs / Hotspots



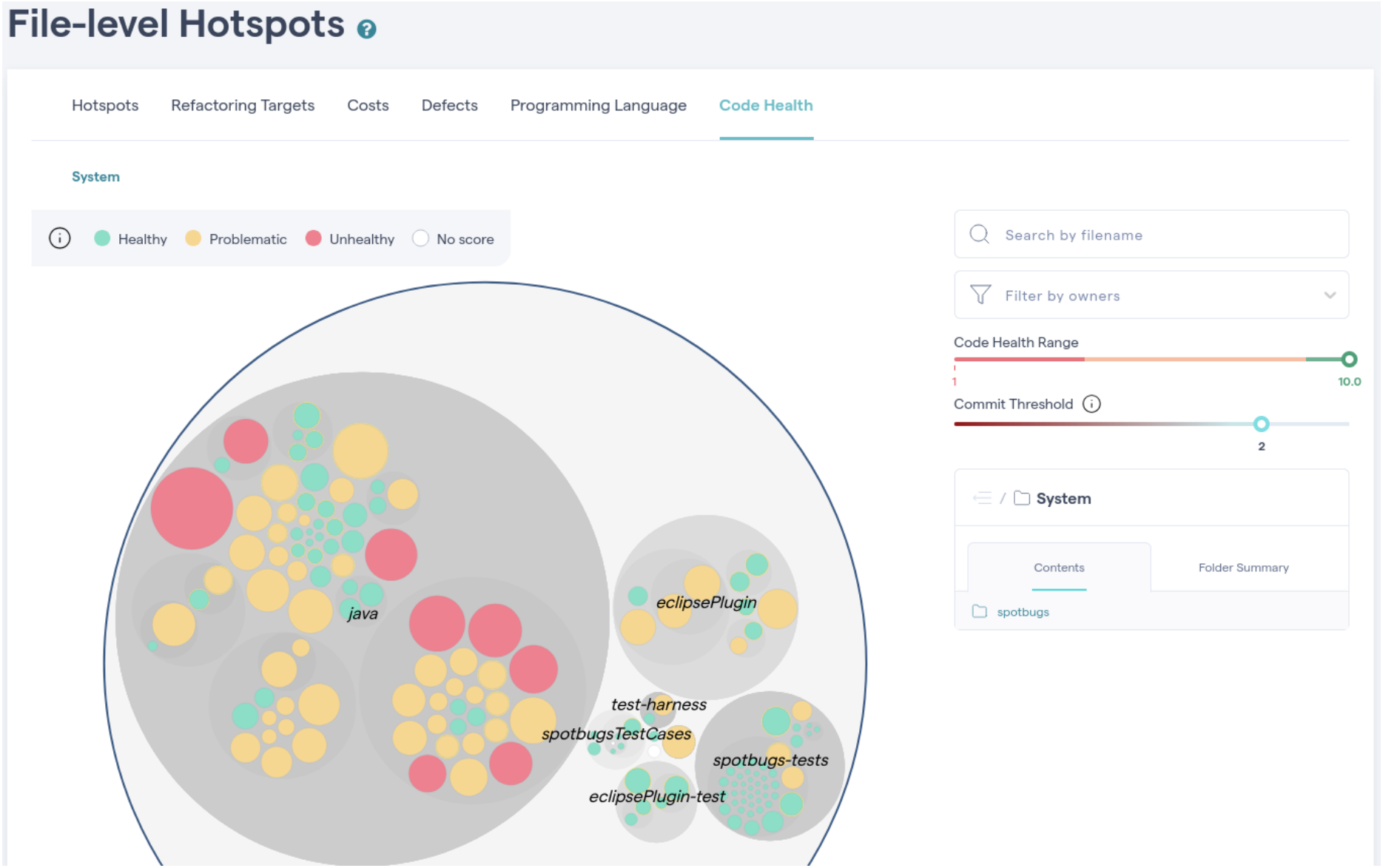
Spotbugs / Hotspots



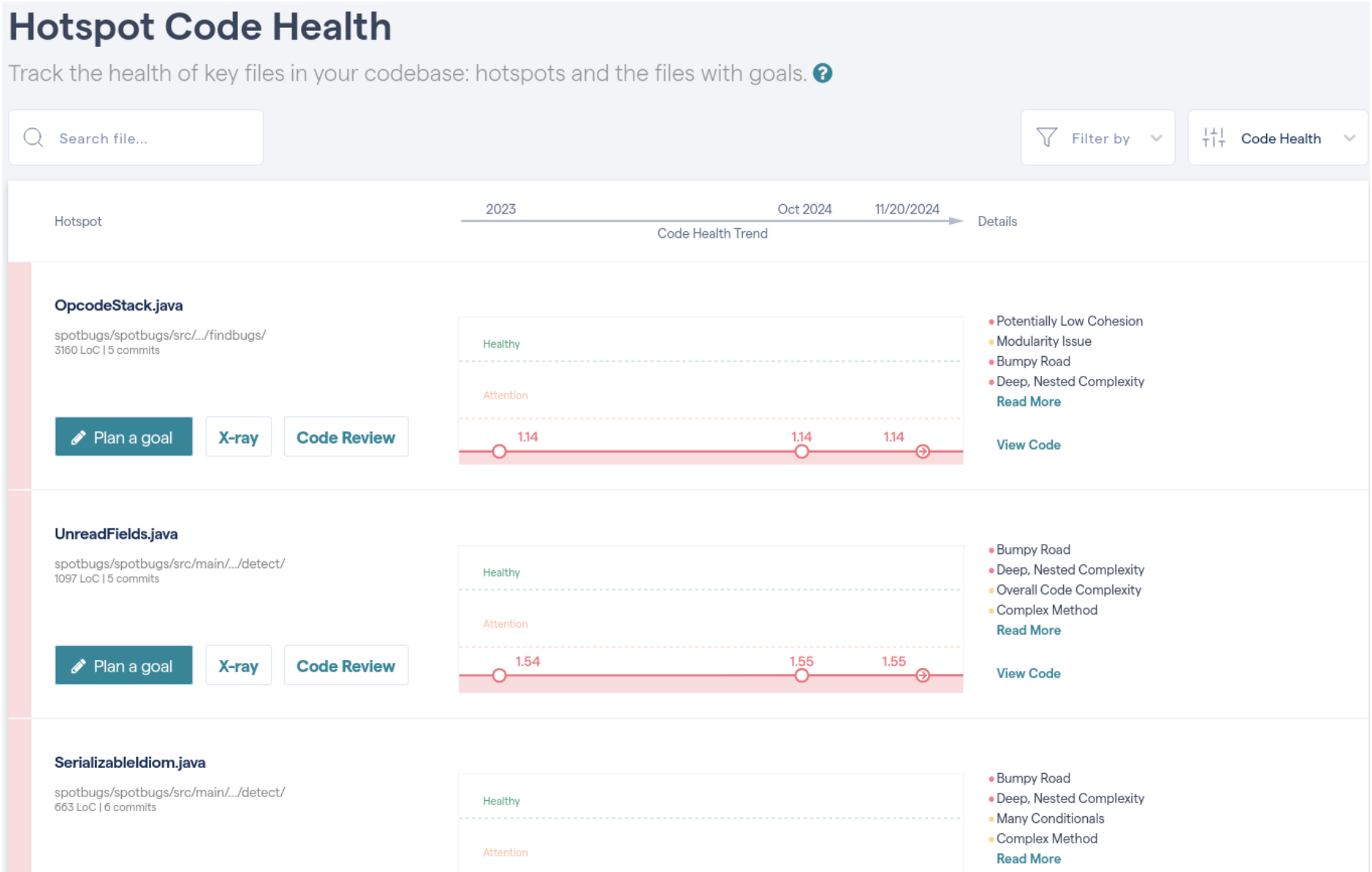
Spotbugs / Hotspots



Spotbugs / Hotspots



Spotbugs / Hotspots / Code Health



Spotbugs / Hotspots / Code Health

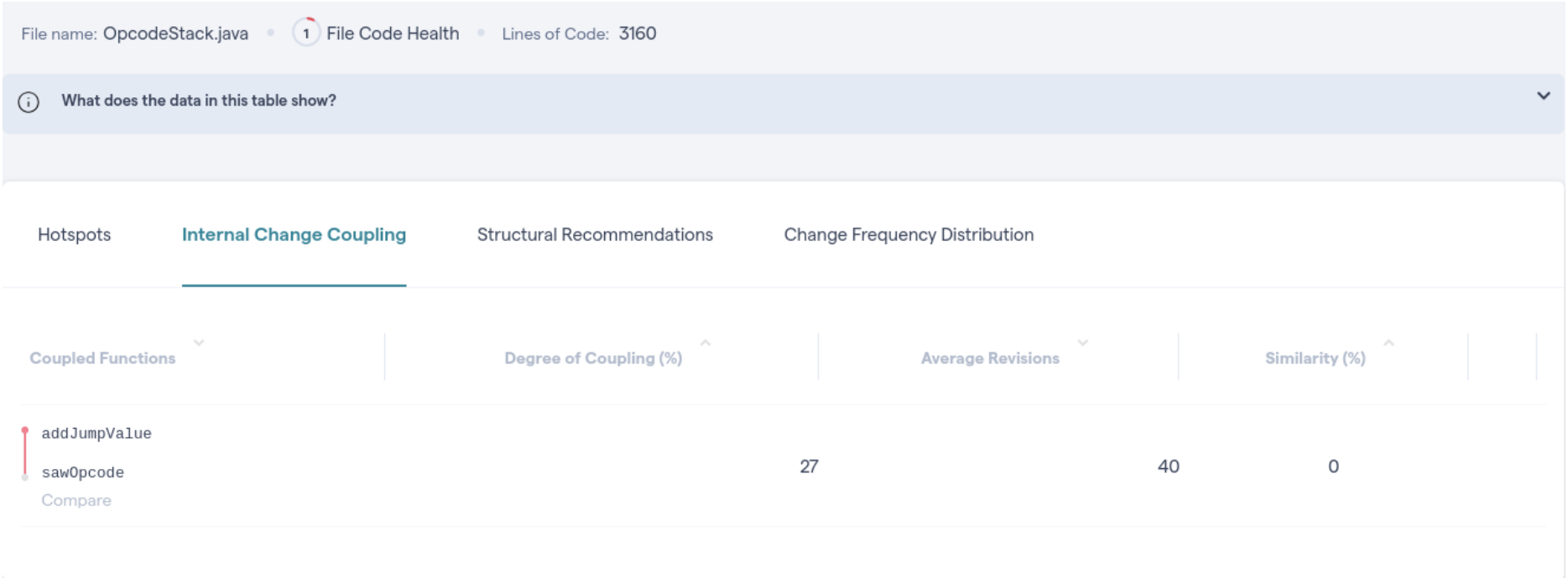
File name: OpcodeStack.java • 1 File Code Health • Lines of Code: 3160

What does the data in this table show?

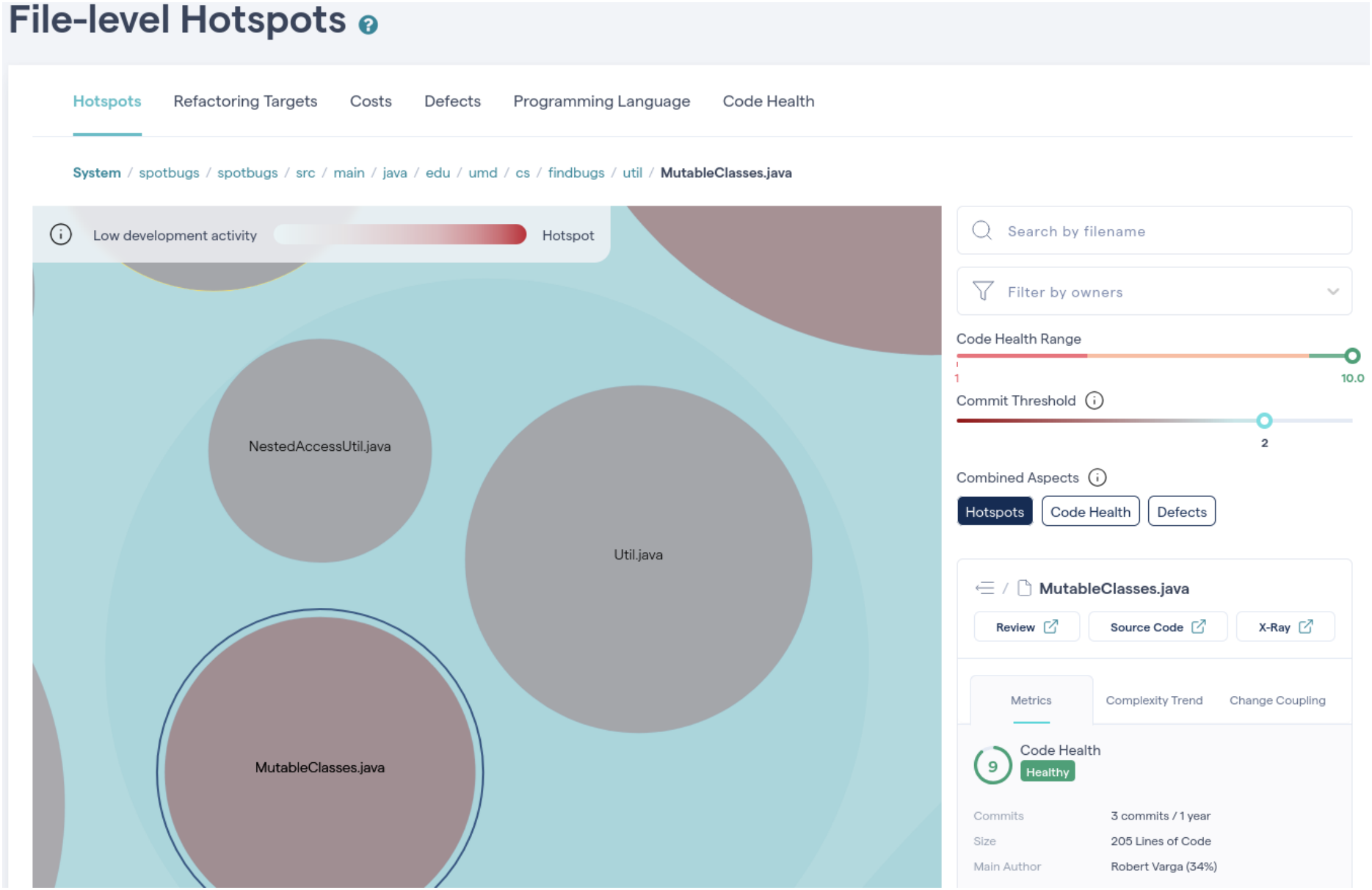
Hotspots | Internal Change Coupling | Structural Recommendations | Change Frequency Distribution

Function name	Change Frequency	Lines of Code	Code Review	Function Complexity	Actions
sawOpcode	65	888	• Complex Method • Bumpy Road Ahead • Deep, Nested Complexity • Complex Conditional	301	⋮
processMethodCall	46	307	• Complex Method • Bumpy Road Ahead • Deep, Nested Complexity • Complex Conditional	135	⋮
pushByIntMath	15	151	• Complex Method • Bumpy Road Ahead • Complex Conditional	73	⋮
OpcodeStack.Item.toString	20	121	• Complex Method • Bumpy Road Ahead	36	⋮
pushByLongMath	6	68	• Complex Method • Bumpy Road Ahead • Deep, Nested Complexity • Complex Conditional	32	⋮

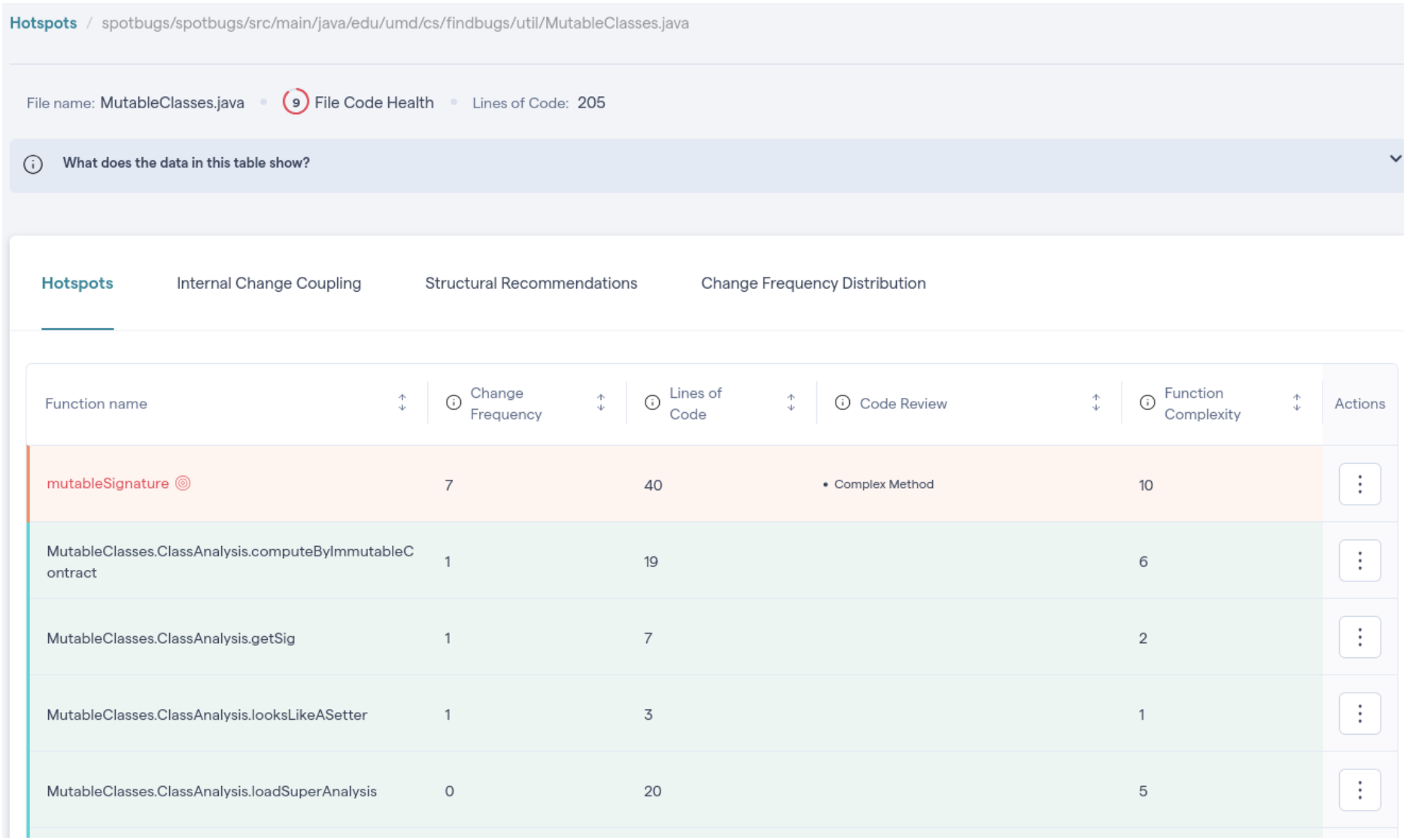
Spotbugs / Hotspots / Code Health



Spotbugs / Hotspots / Code Health



Spotbugs / Hotspots / Code Health



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

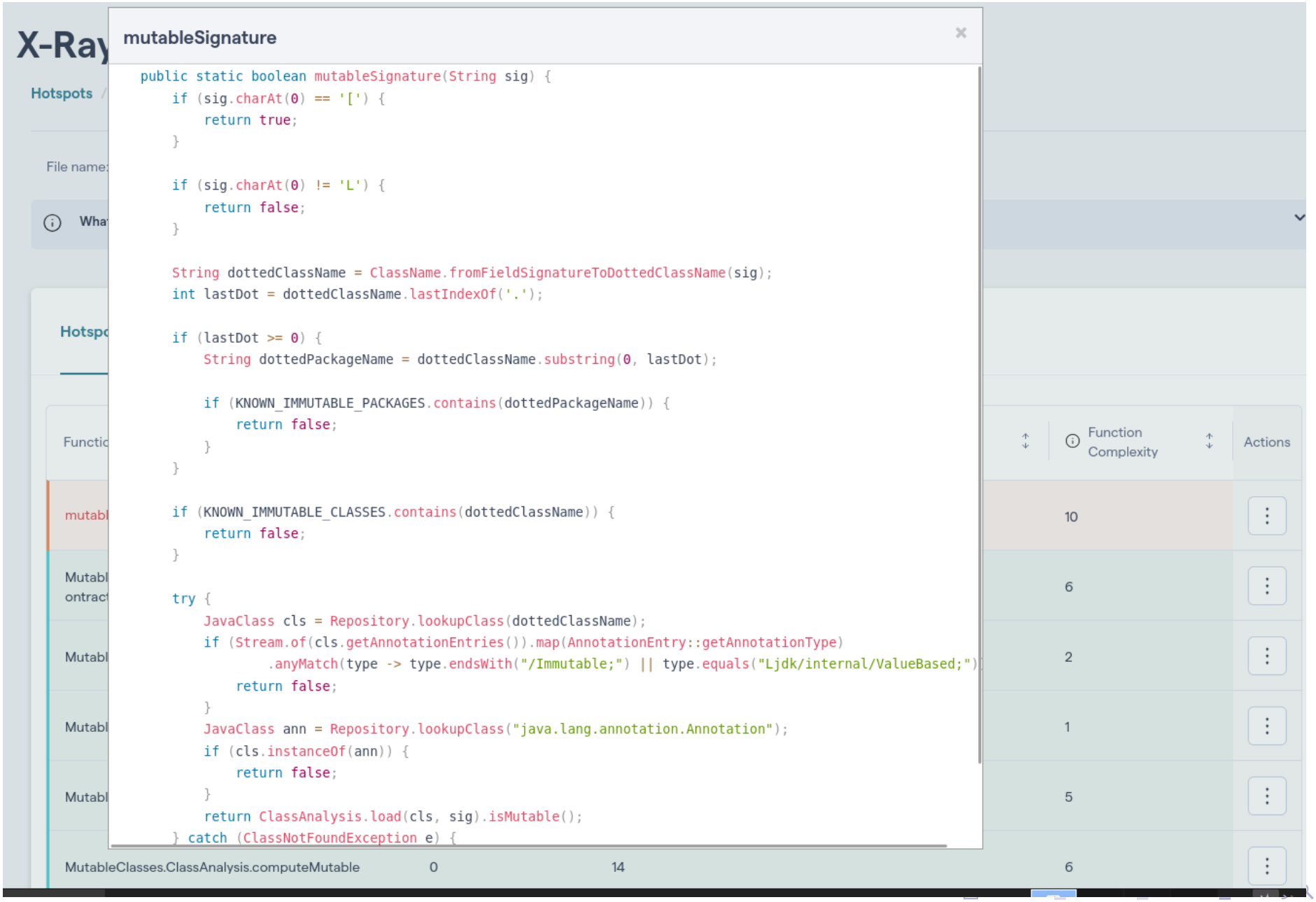
File name: MutableClasses.java • 9 File Code Health • Lines of Code: 205

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



Spotbugs / Hotspots / Change coupling

By Commits | Across Commits | Between Repositories | Sum of Couplings

Minimum Degree of Coupling: 70%

Filter temporal coupling below the given percentage.

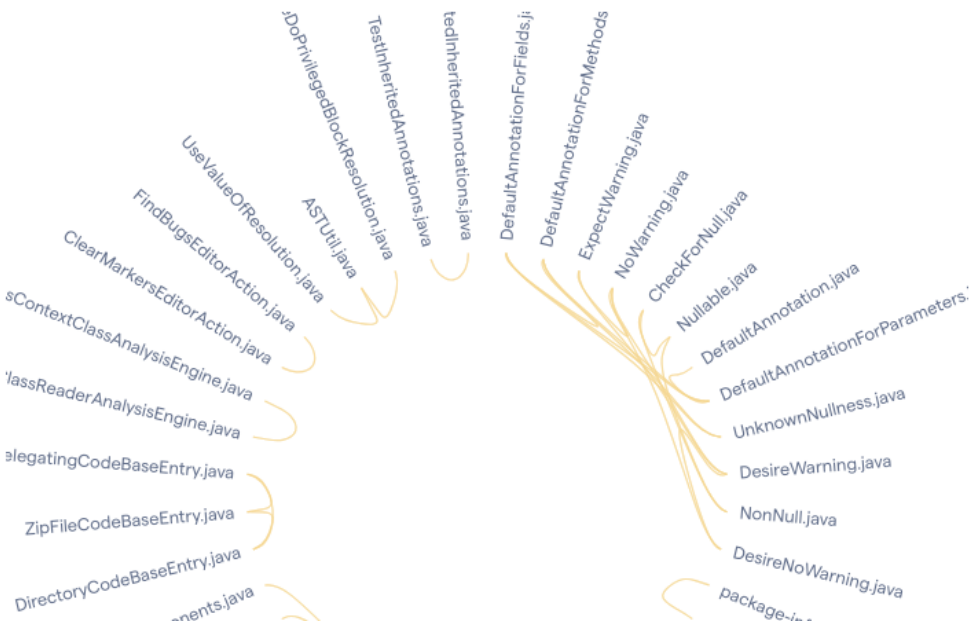
Select overlay: None

Exclusion filters

C/C++ Includes | Java Impls | C# Interfaces | Java Tests | Python Tests | C# Tests

Month | Year

Stable | Increasing | Decreasing



Coupled Entities	Degree of coupling	Average Revisions
spotbugs/spotbugs-annotations/src/main/java/edu/umd/cs/findbugs/annotations/DefaultAnnotationForFields.java spotbugs/spotbugs-annotations/src/main/java/edu/umd/cs/findbugs/annotations/DefaultAnnotationForMethods.java	100%	7
spotbugs/spotbugs-annotations/src/main/java/edu/umd/cs/findbugs/annotations/ExpectWarning.java spotbugs/spotbugs-annotations/src/main/java/edu/umd/cs/findbugs/annotations/NoWarning.java	96%	17
spotbugs/spotbugs-annotations/src/main/java/edu/umd/cs/findbugs/annotations/CheckForNull.java spotbugs/spotbugs-annotations/src/main/java/edu/umd/cs/findbugs/annotations/Nullable.java	93%	8
spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/ba/obl/package-info.java spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/ba/package-info.java	90%	6
spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/filter/FieldMatcher.java spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/filter/MethodMatcher.java	89%	15
spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/graph/MergeVertices.java spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/graph/Transpose.java	88%	9
spotbugs/spotbugs/src/main/java/edu/umd/cs/findbugs/classfile/impl/DirectoryCodeBaseEntry.java		

Spotbugs / Architecture / Recover

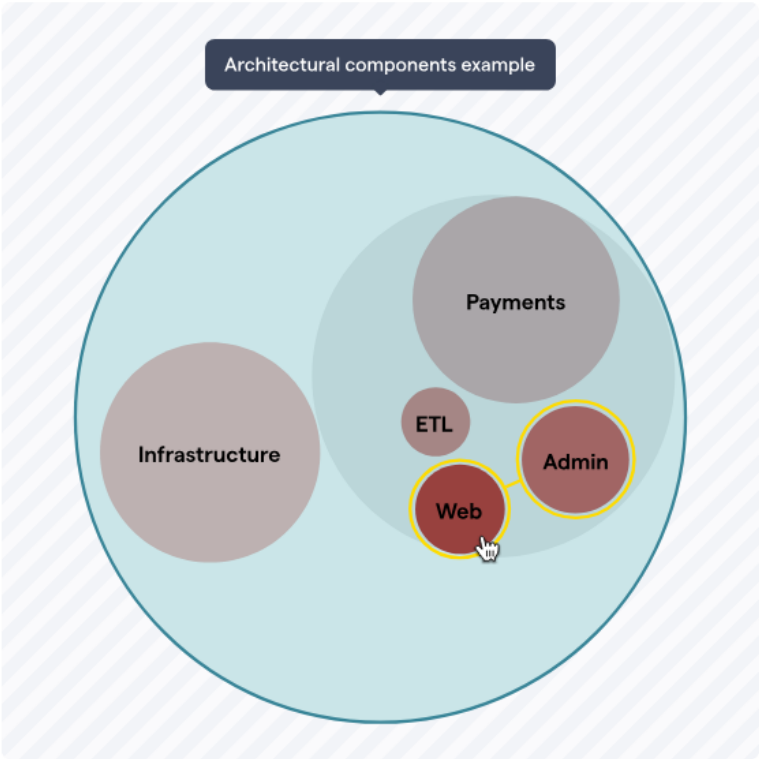
Architectural analyses

Architectural components are not configured

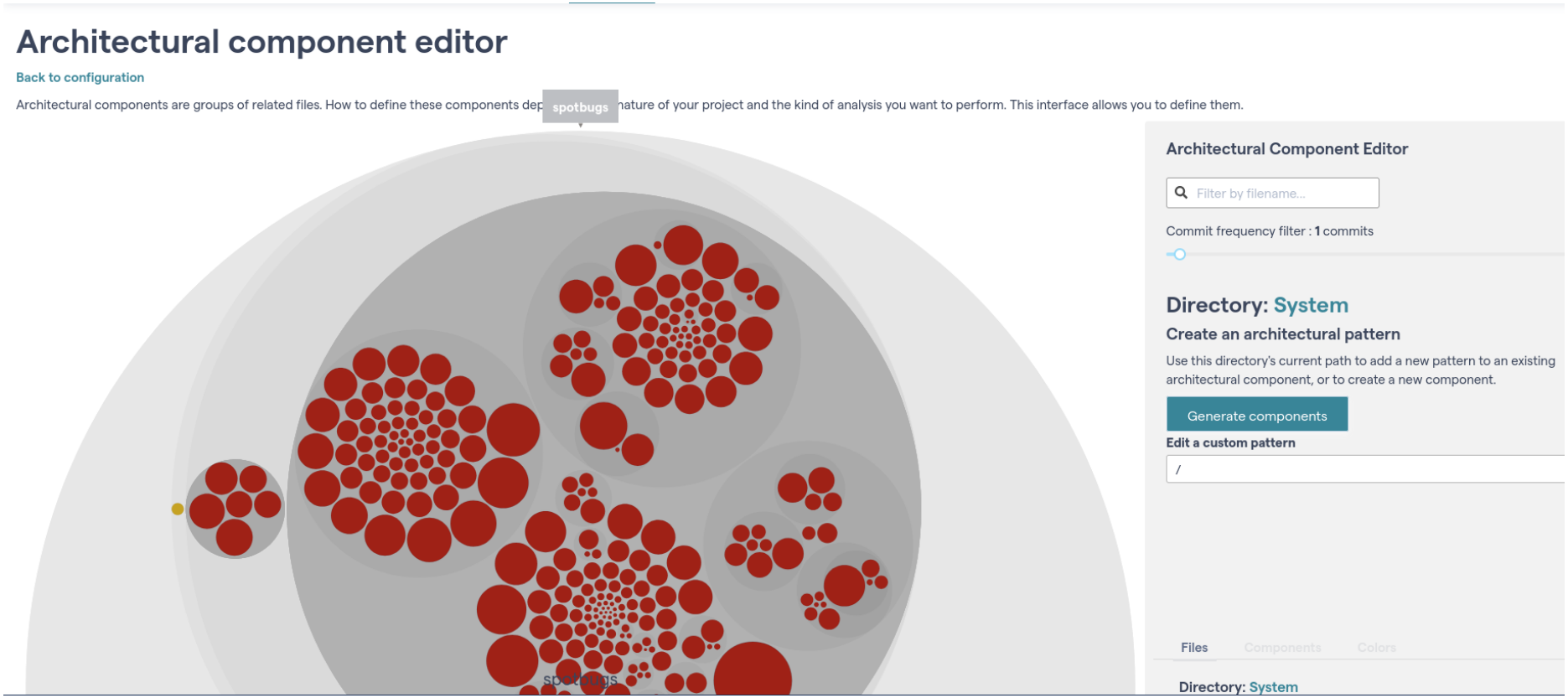
With CodeScene's Architectural Components, you can:

- **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.

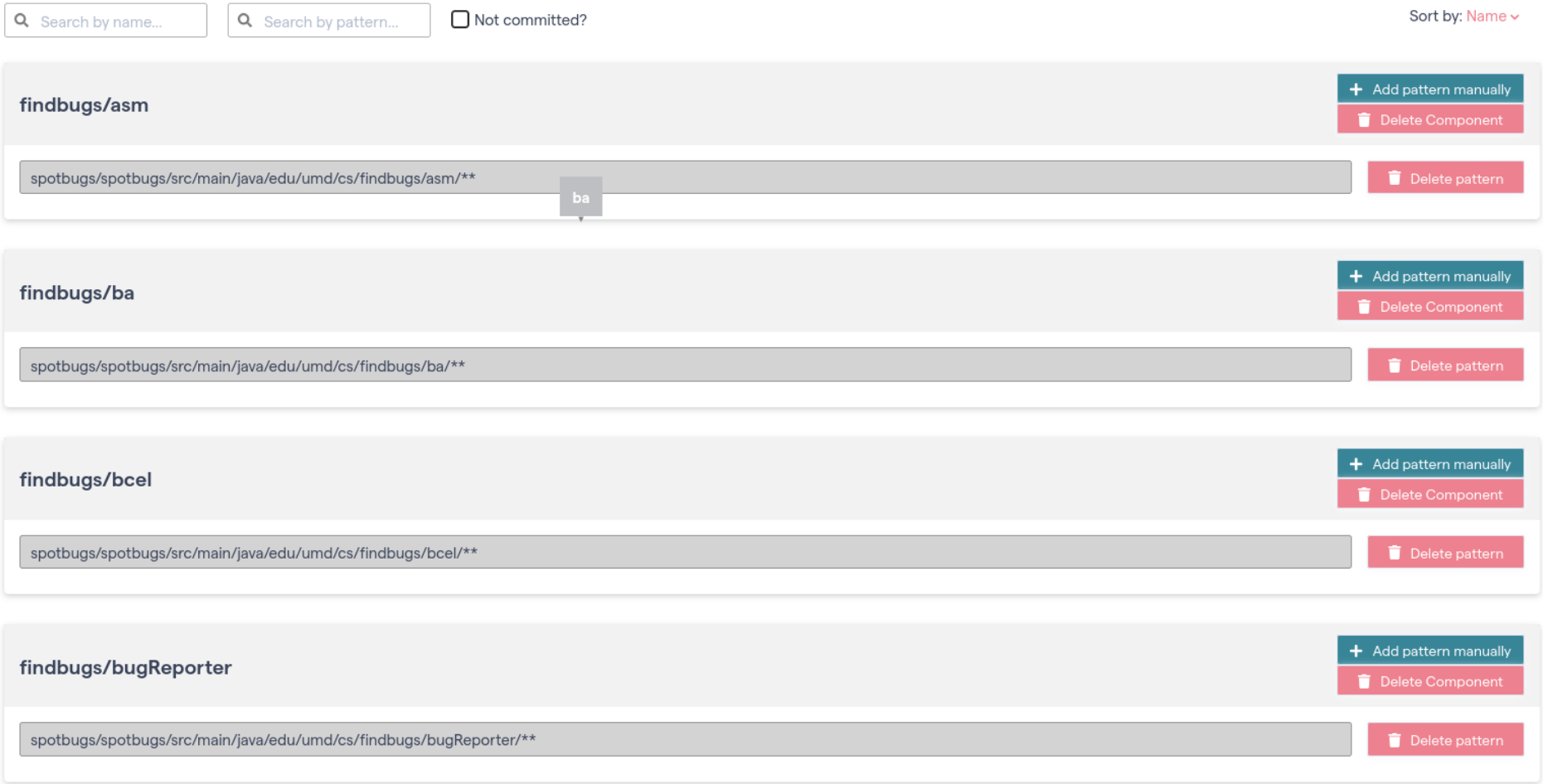
Setup Architectural Components



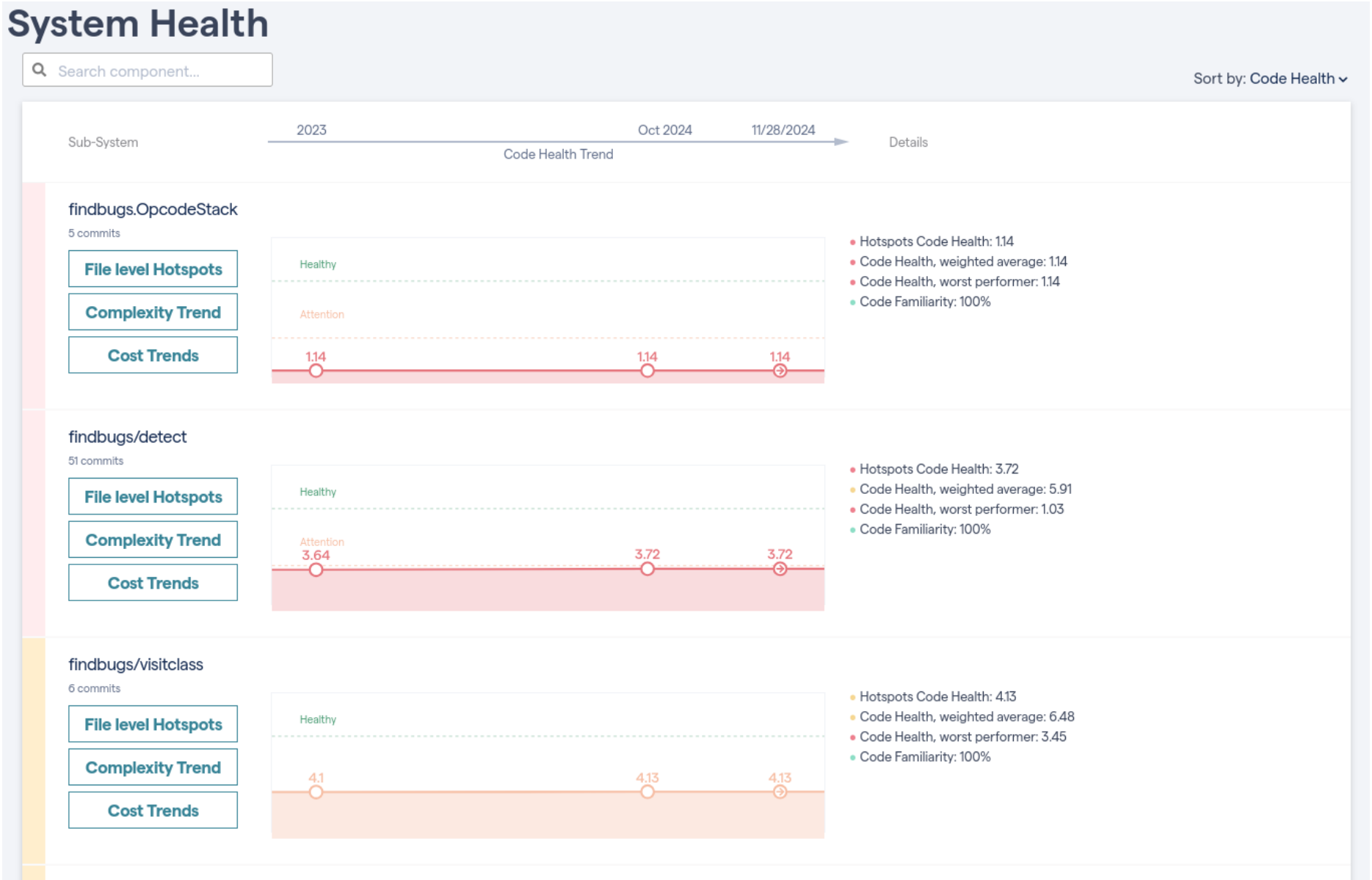
Spotbugs / Architecture / Recover



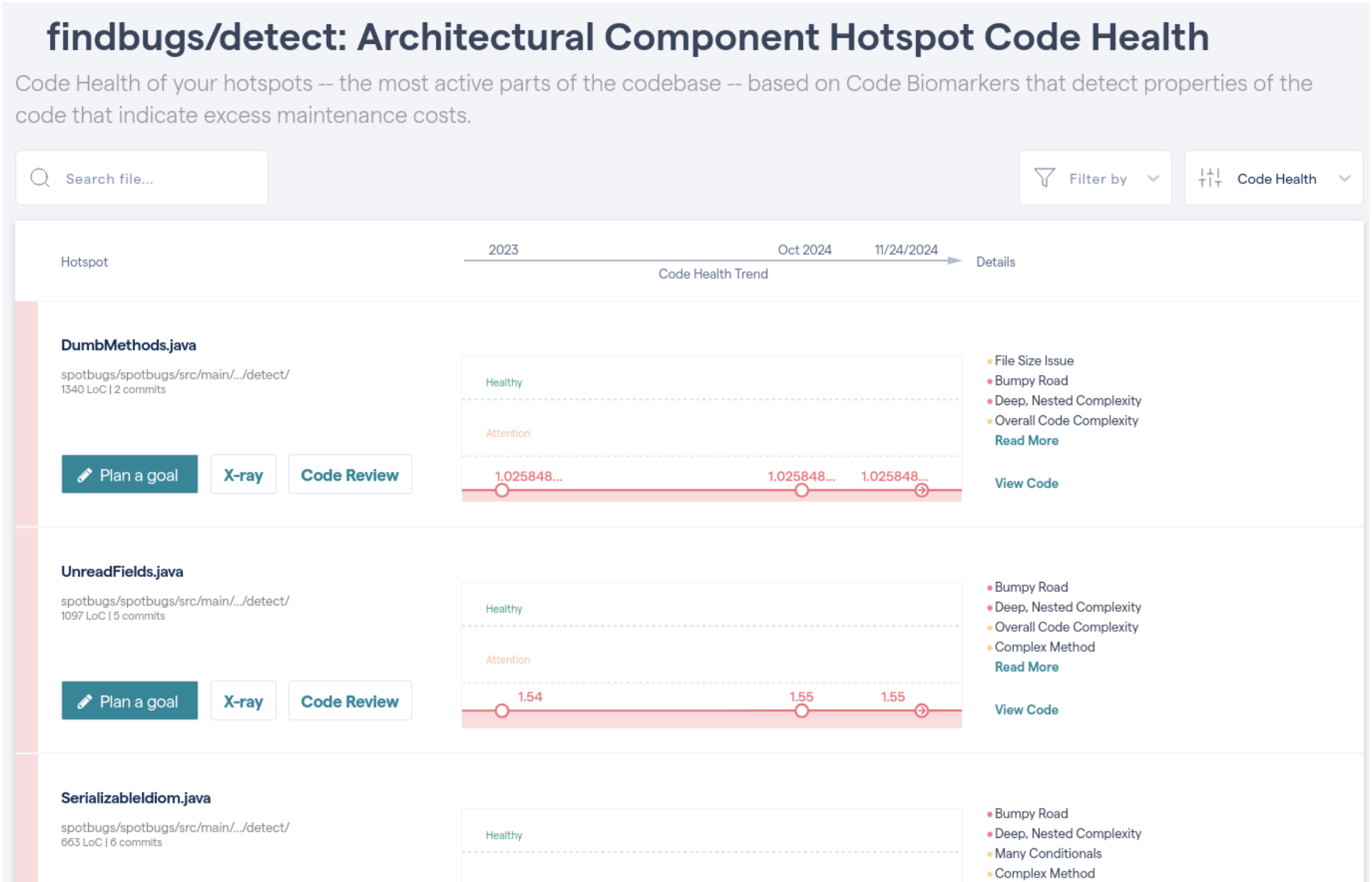
Spotbugs / Architecture / Recover



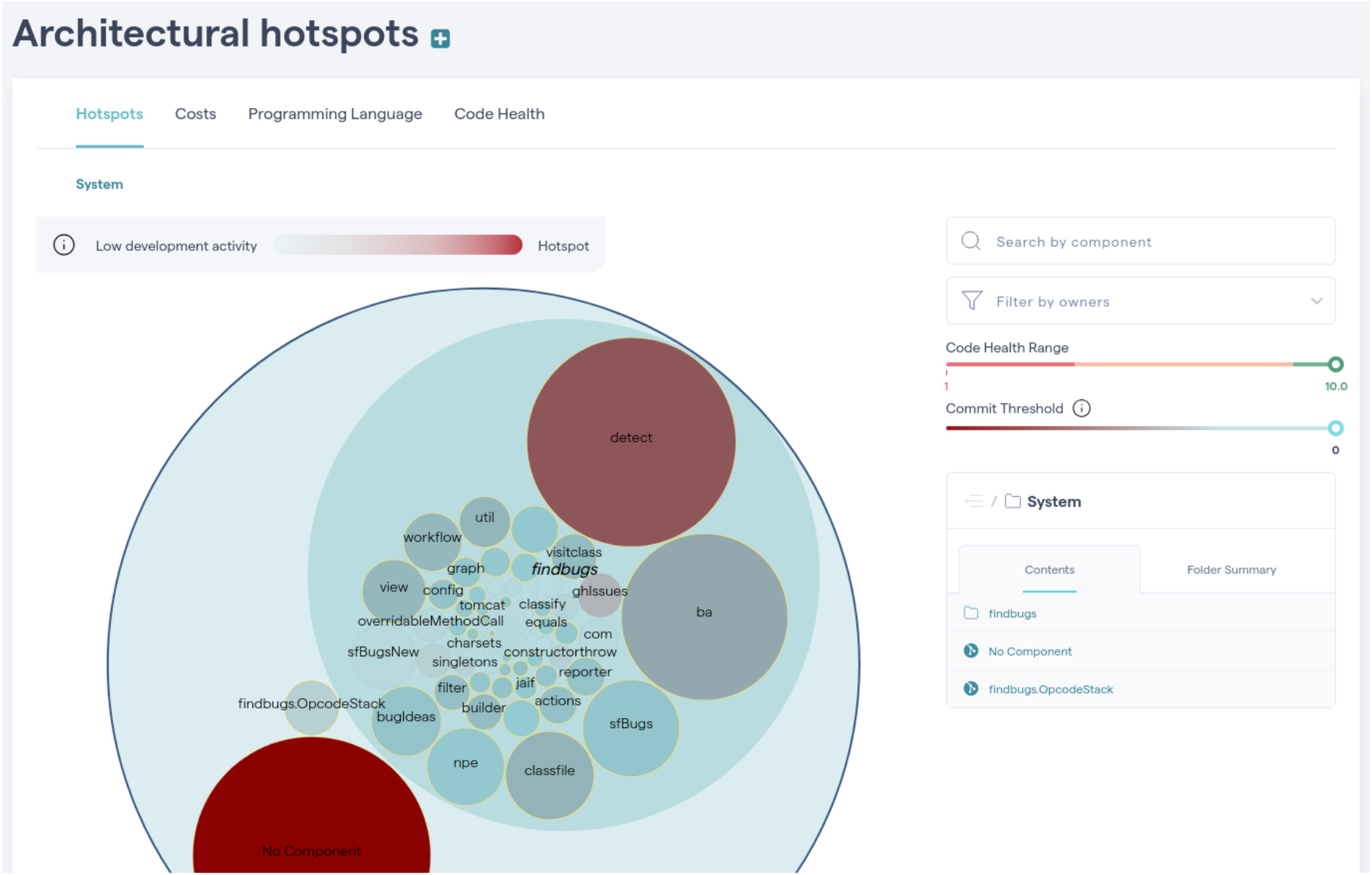
Spotbugs / Architecture / Overview



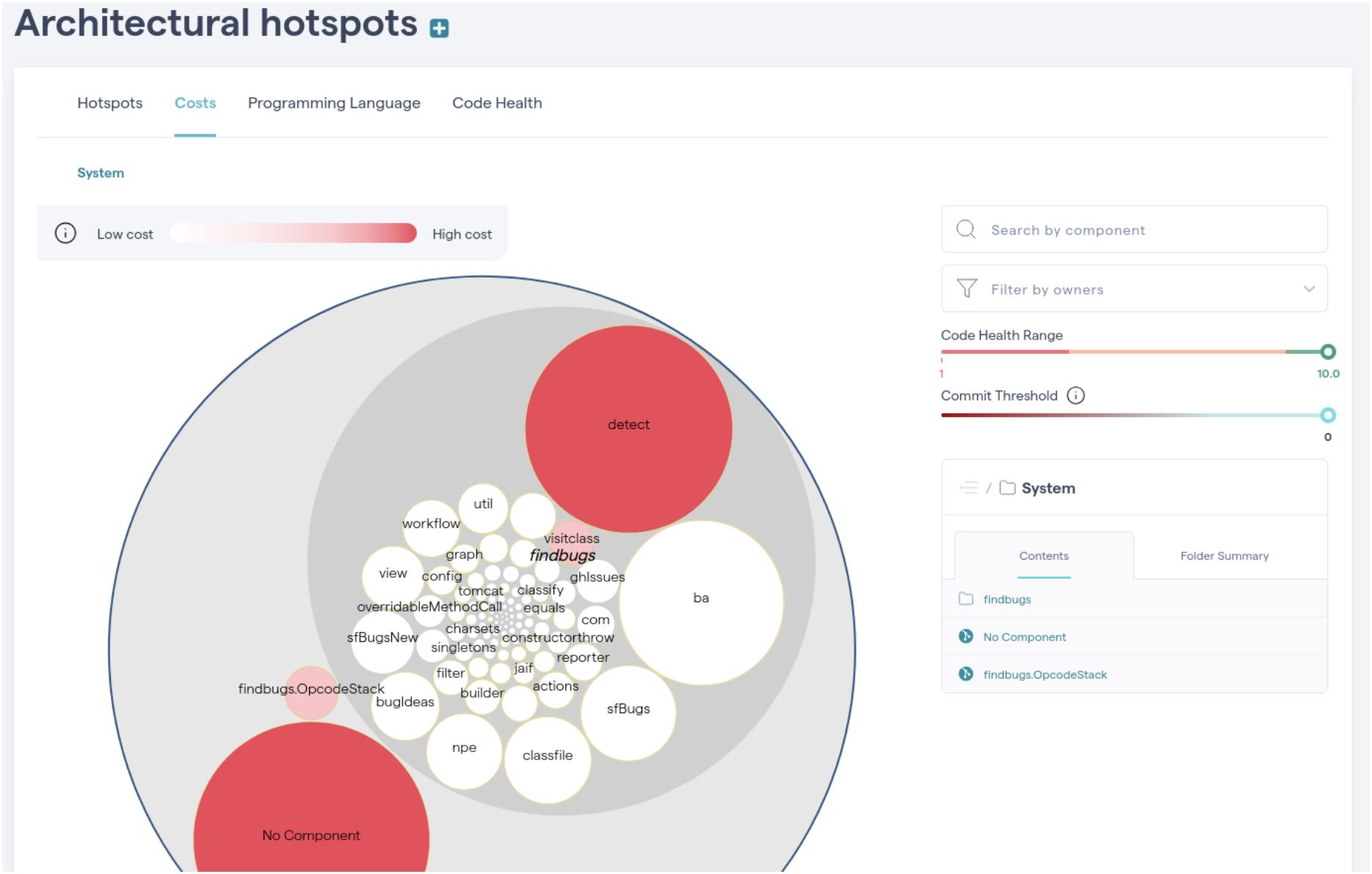
Spotbugs / Architecture / Details



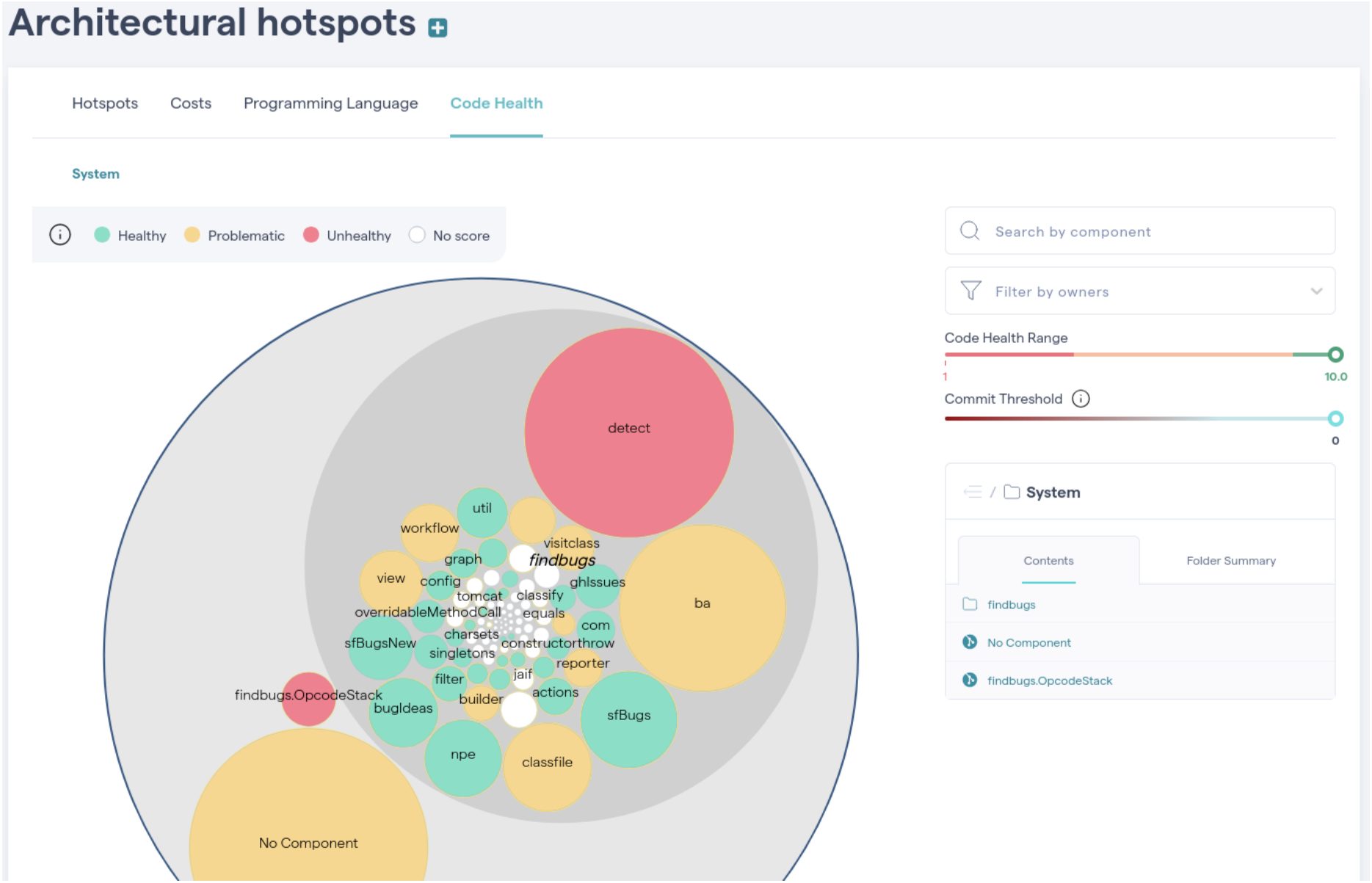
Spotbugs / Architecture / Hotspots



Spotbugs / Architecture / Costs



Spotbugs / Architecture / Code Health

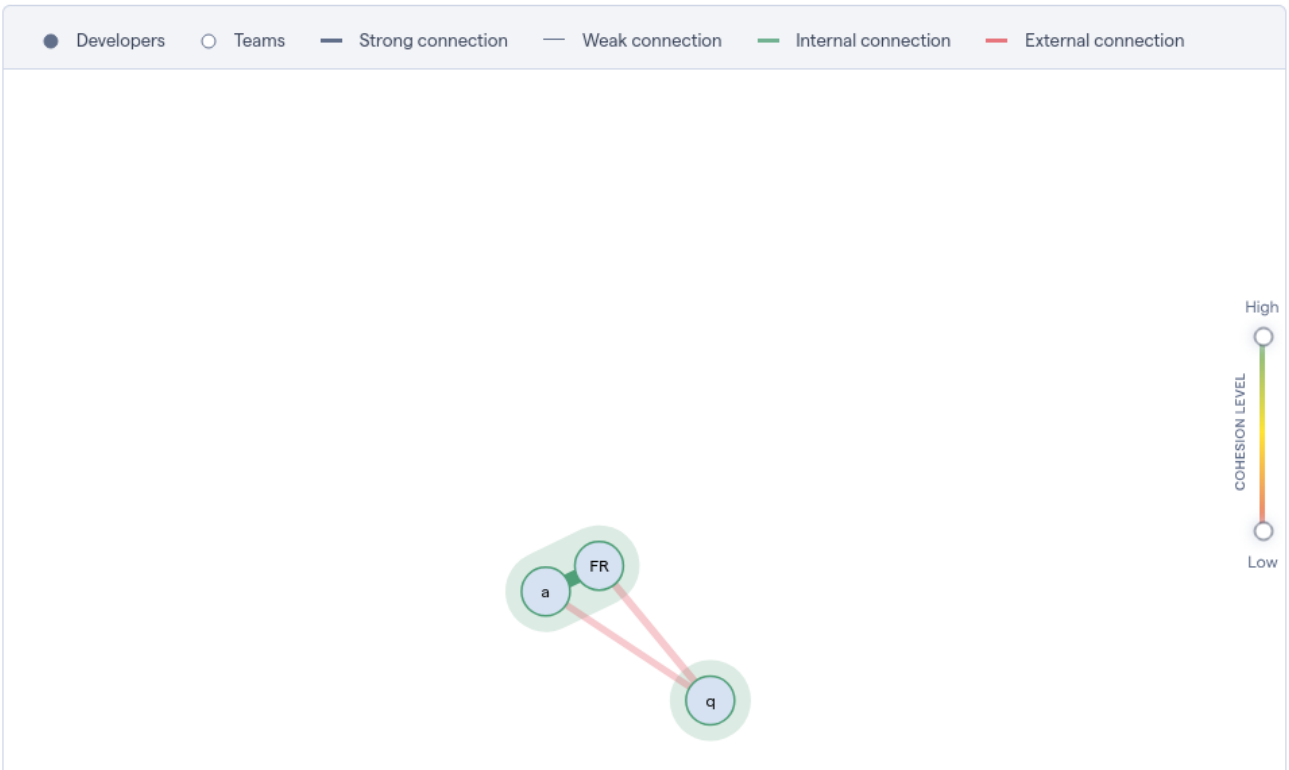


Brewday / Teams

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?

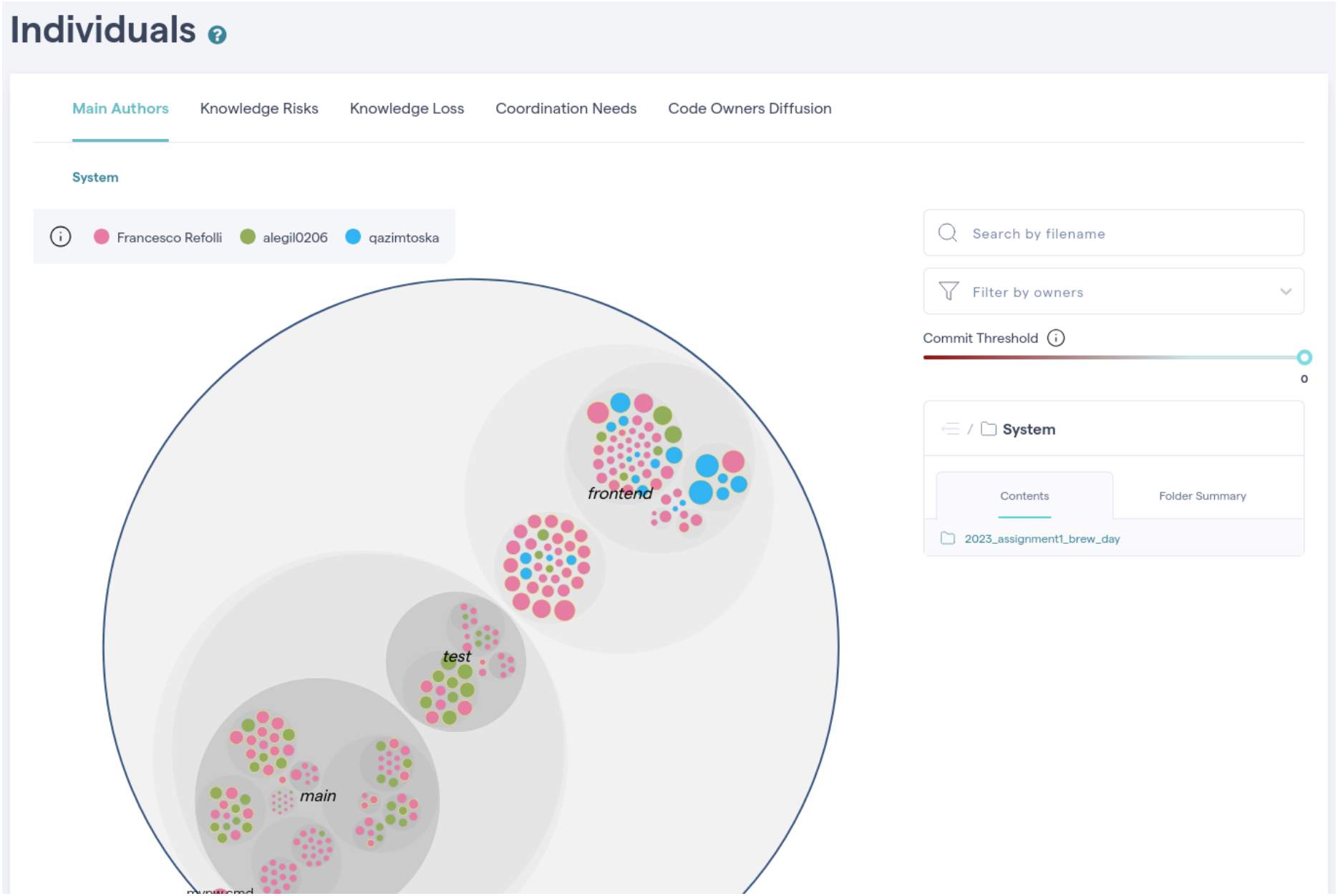


Teams > Developers

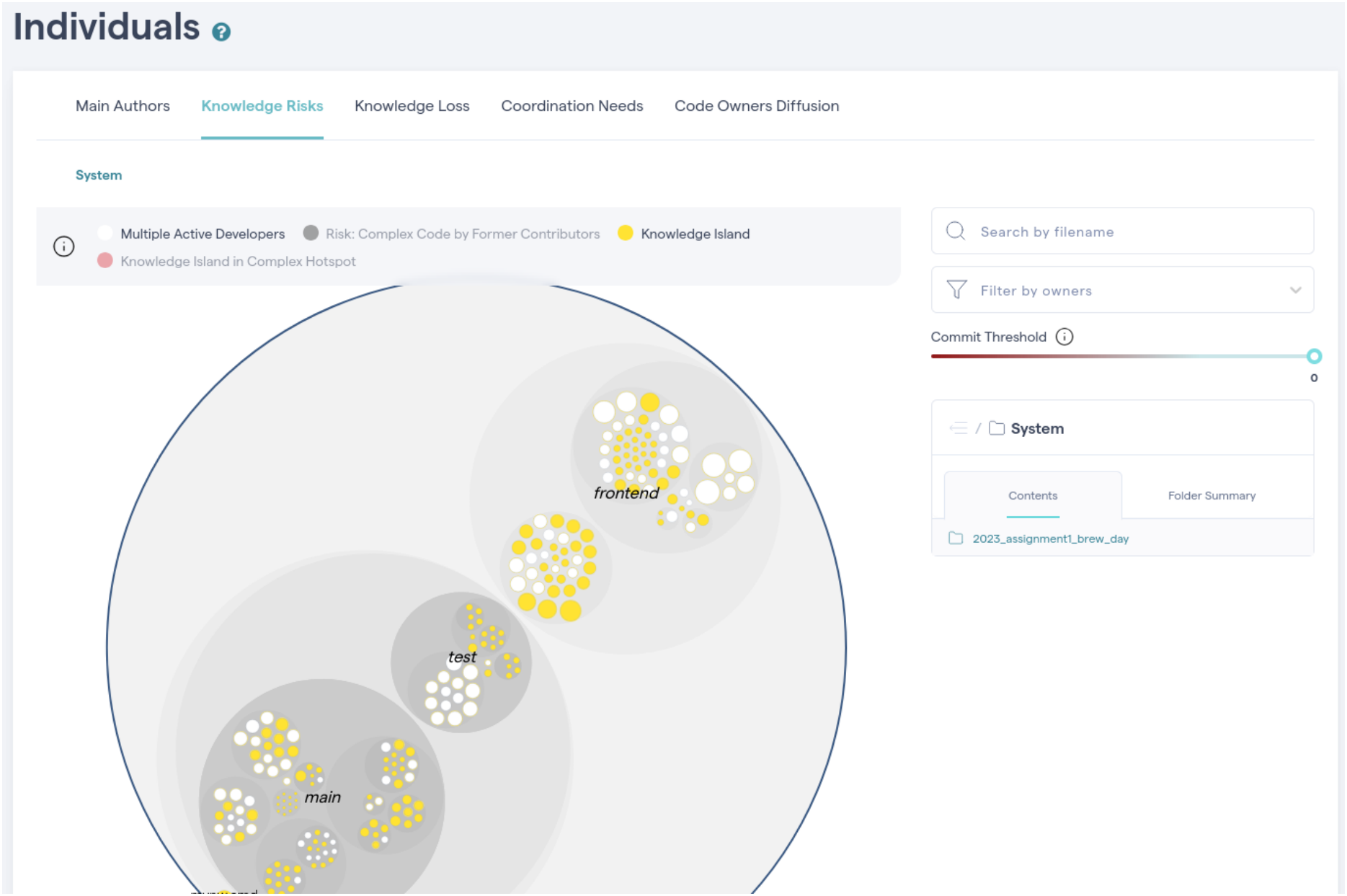
BACKEND (2 DEVELOPERS)

FRONTEND (1 DEVELOPER)

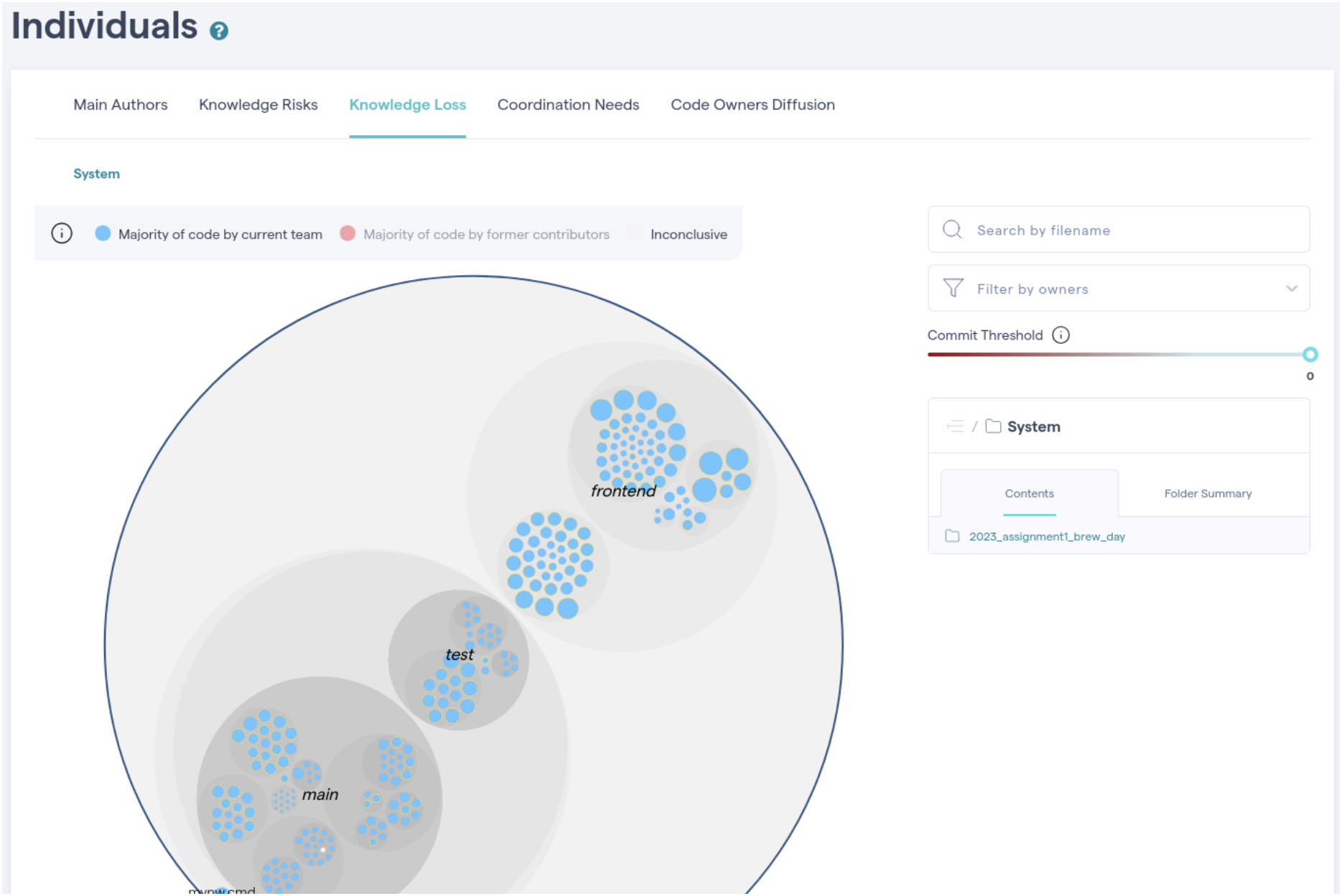
Brewday / Teams



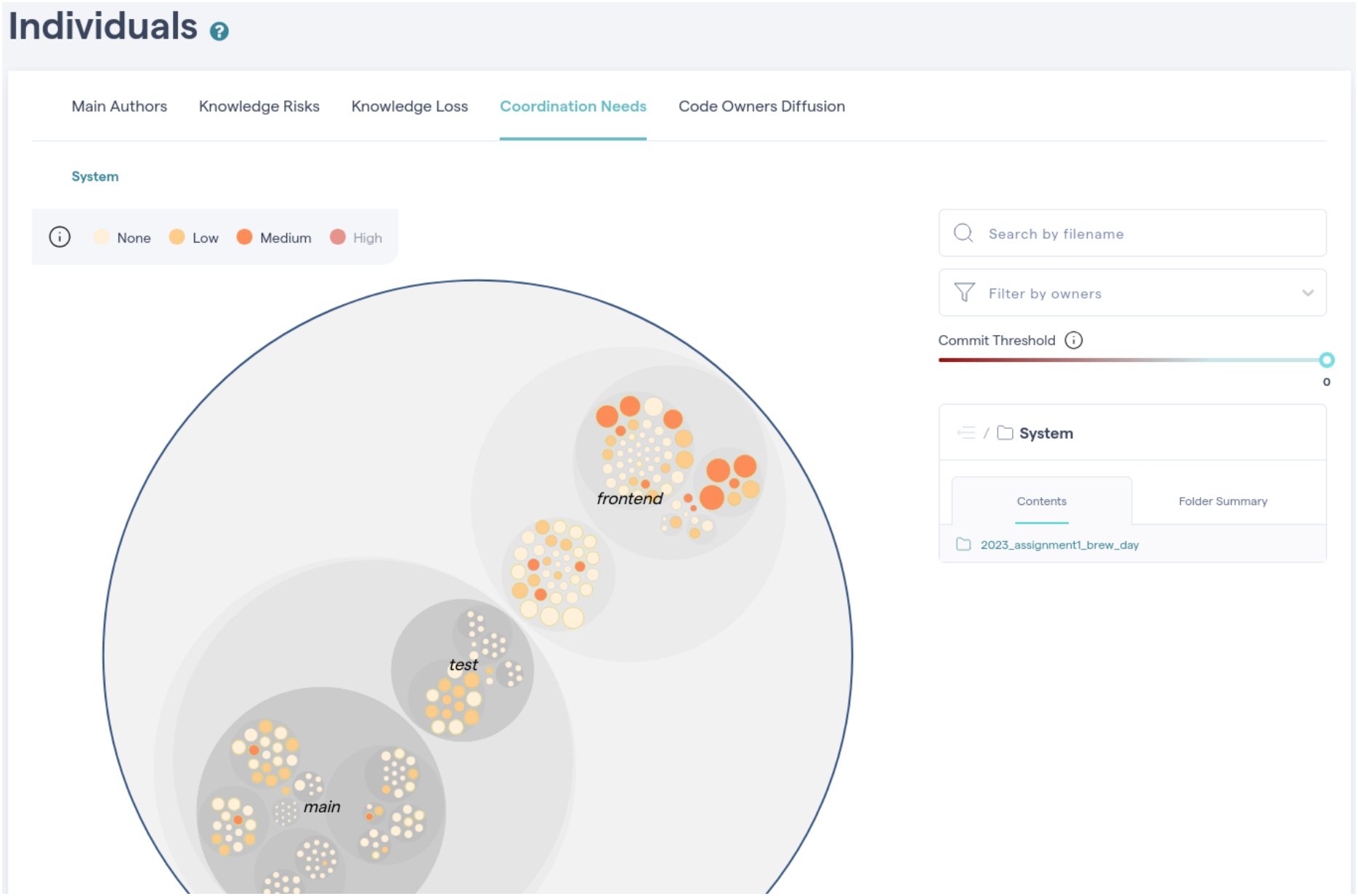
Brewday / Teams



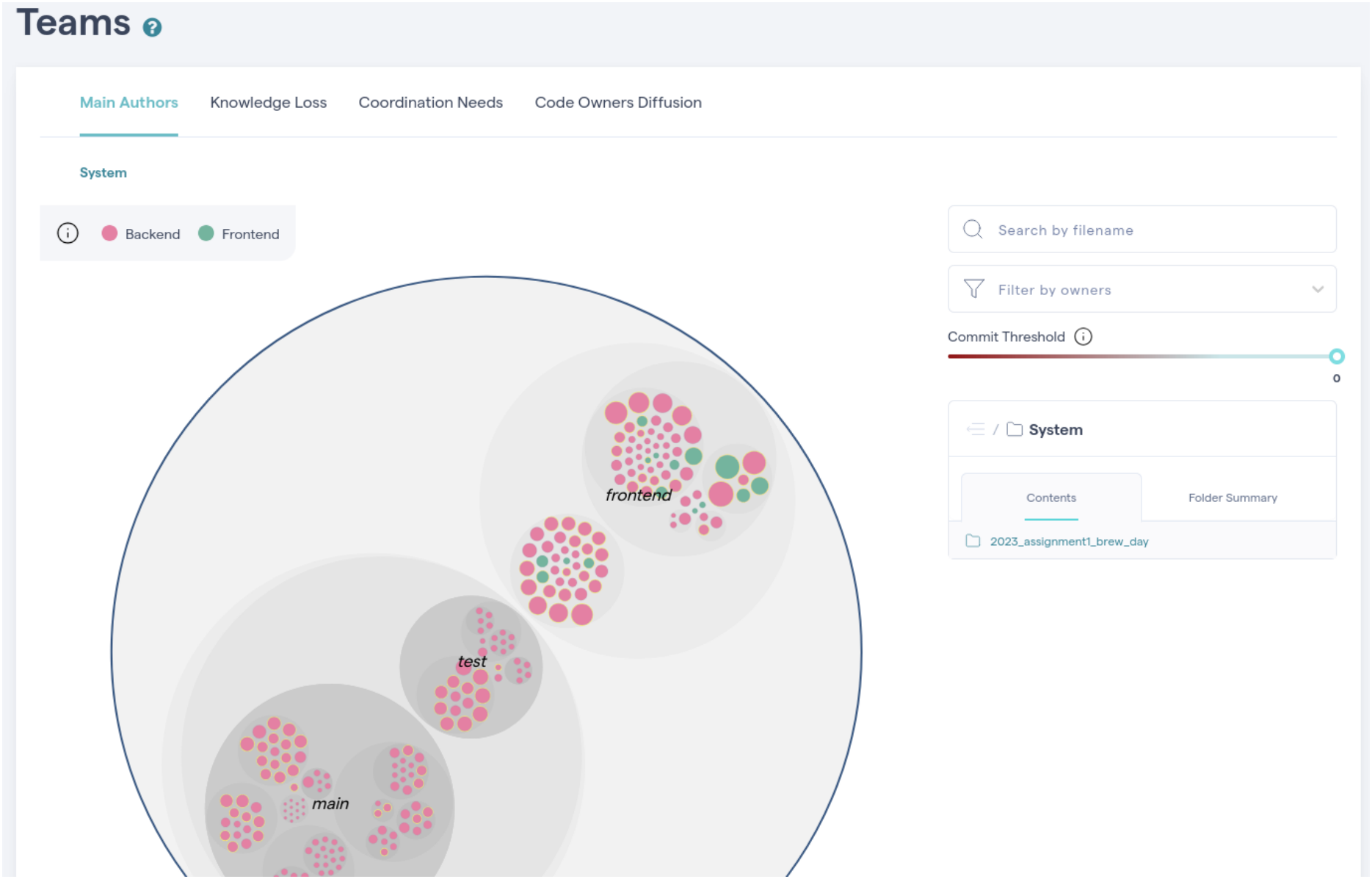
Brewday / Teams



Brewday / Teams



Brewday / 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**.

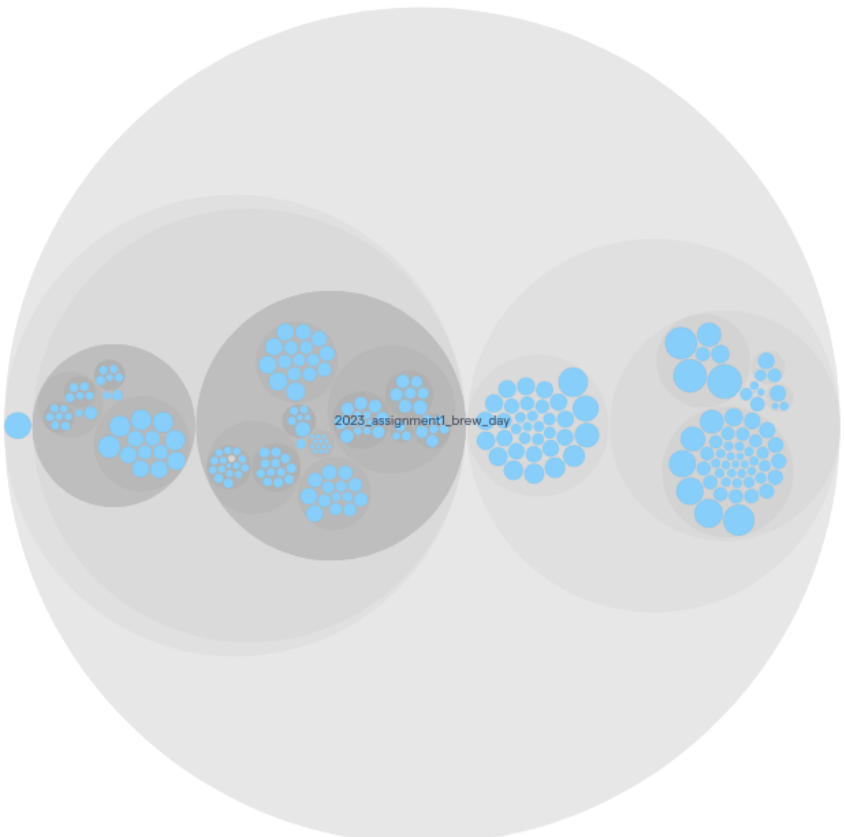
Search author by nam

Authors	Primary Component Owner	Primary File Owner	Commit Pattern	Former Contributor	Last Contribution	Months Contributing	Commits	LOC Added	LOC Removed	LOC Net
Francesco Refolli	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:

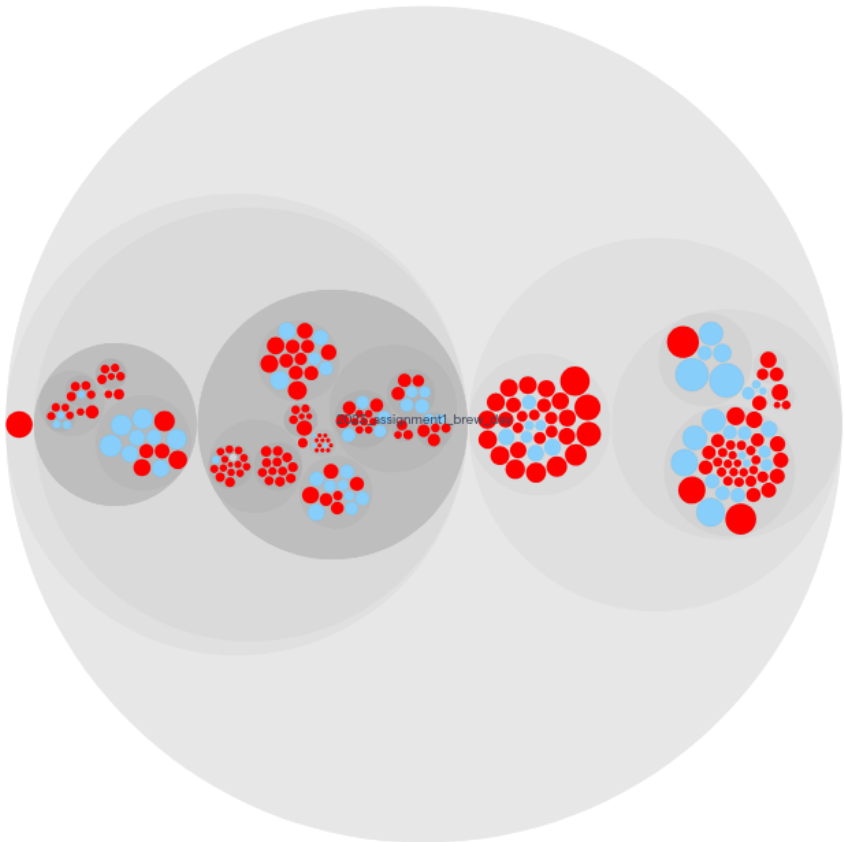
A regex or a string

- 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?



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

- [x] Francesco Refolli

Developers

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

Filter developer names:

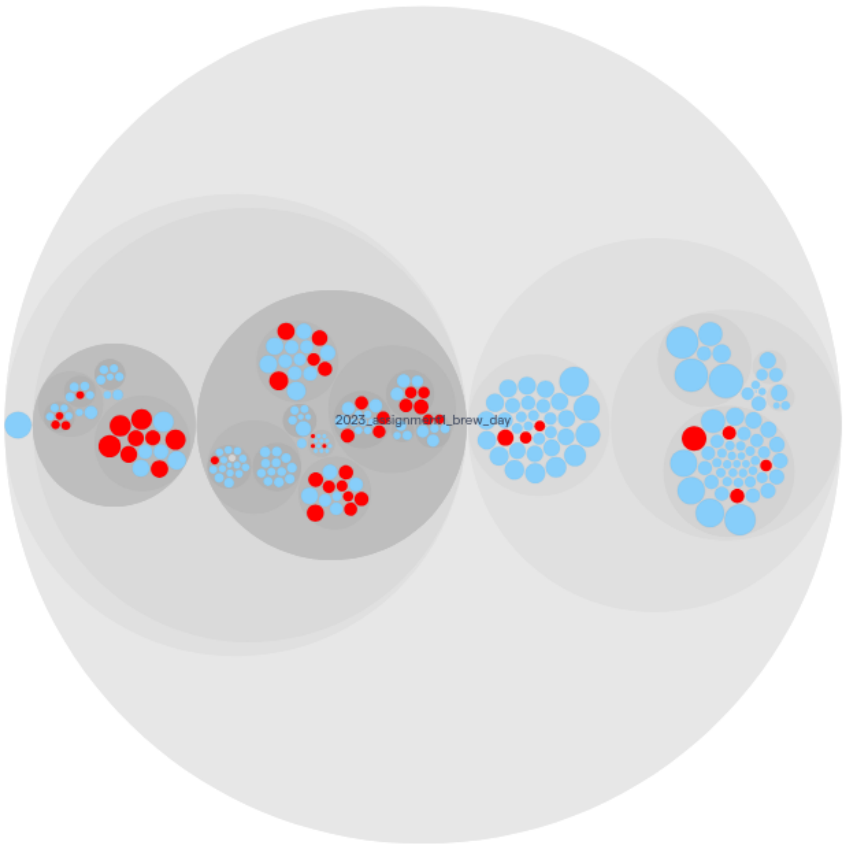
A regex or a string

- [ ] alegil0206
- [x] Francesco Refolli
- [ ] 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

- [x] alegil0206

Developers

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

Filter developer names:

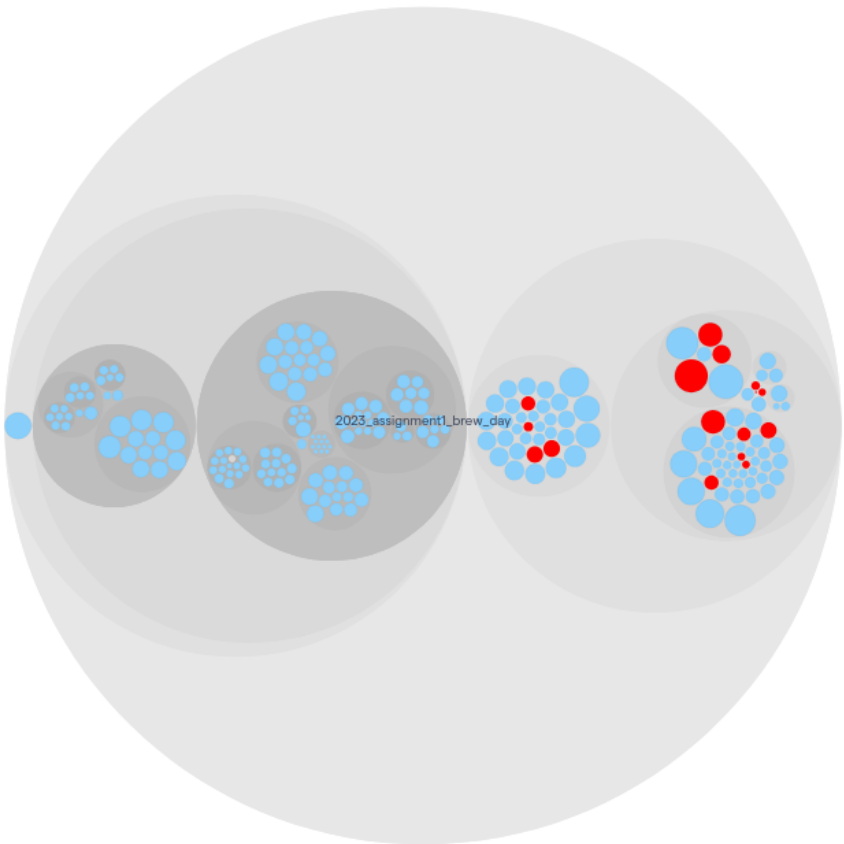
A regex or a string

- [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?



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

- [x] qazimtoska

Developers

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

Filter developer names:

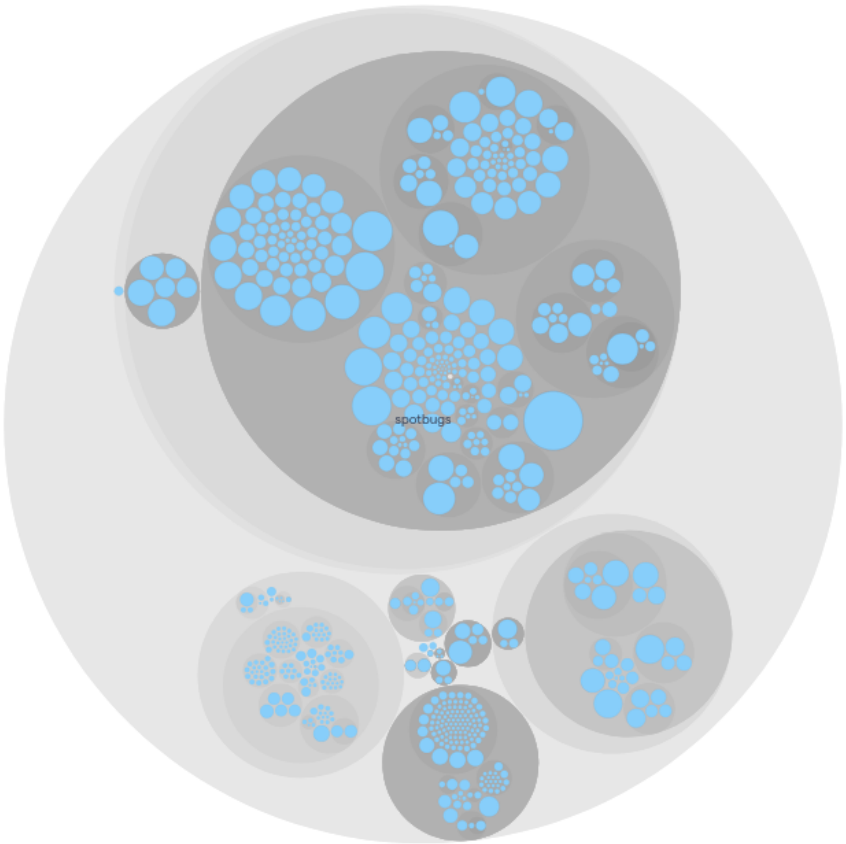
A regex or a string

- [ ] alegil0206
- [ ] Francesco Refolli
- [x] 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

/

Directory: **System**

spotbugs

Developers

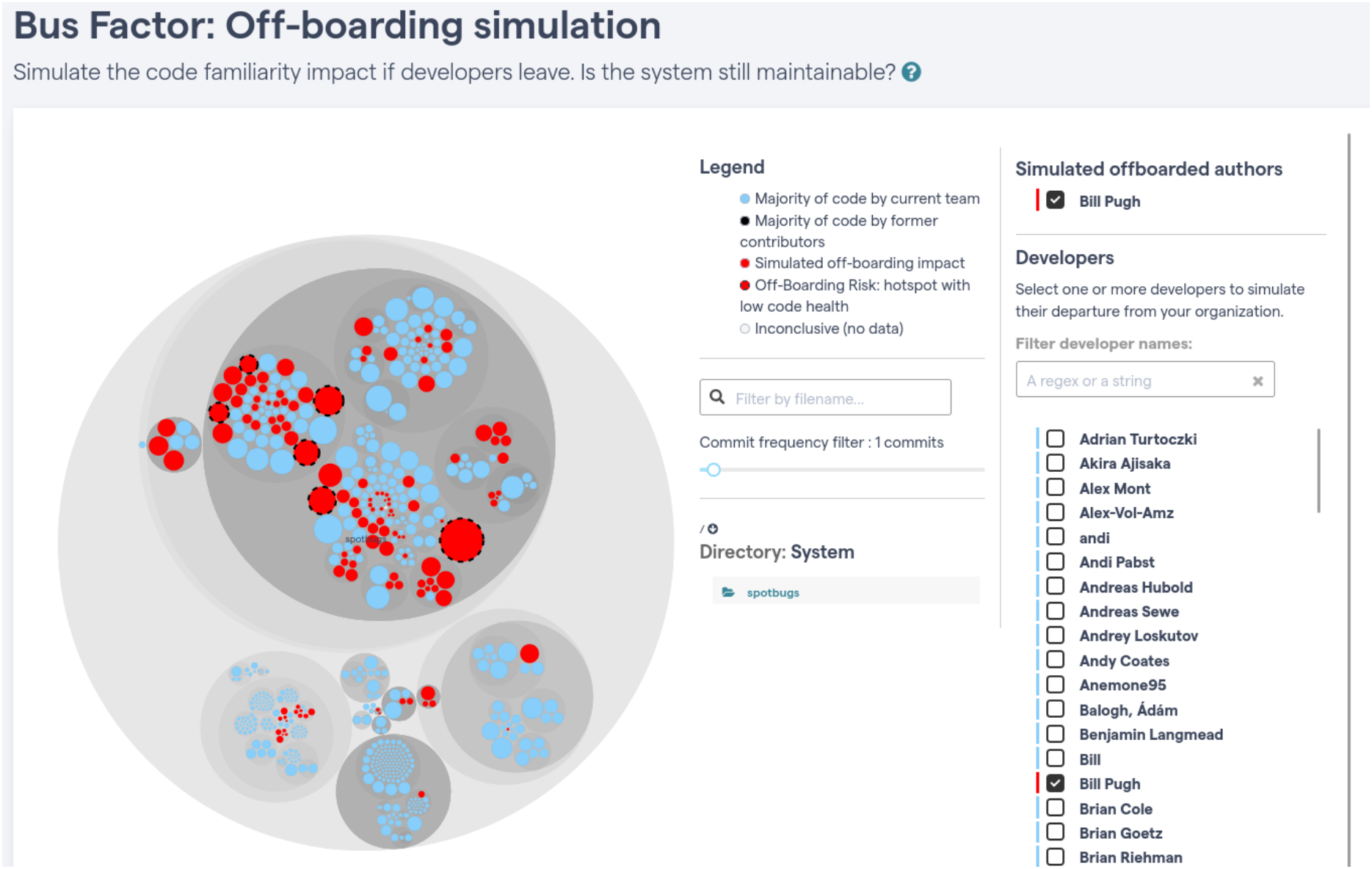
Select one or more developers to simulate their departure from your organization.

Filter developer names:

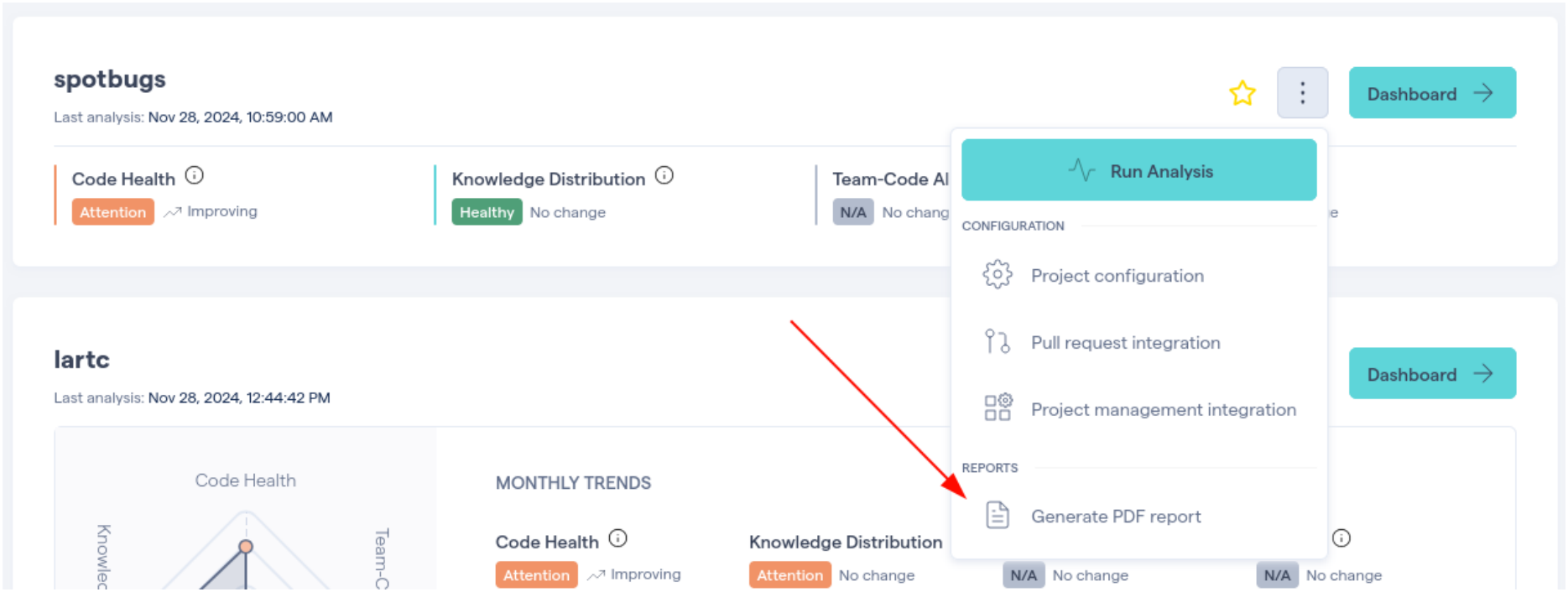
A regex or a string

- Adrian Turtoczki
- Akira Ajisaka
- 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

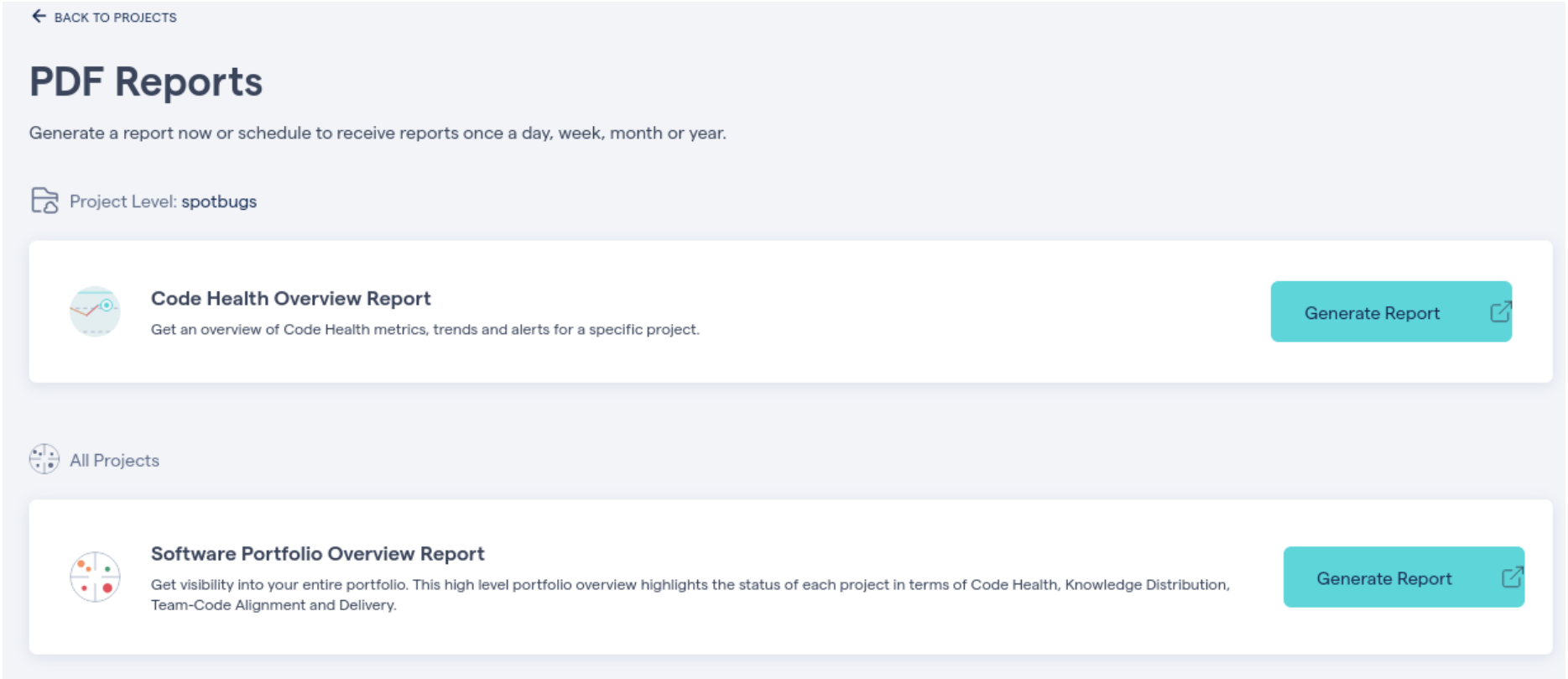
Spotbugs / Simulations / Layoffs



Projects Report



Projects 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?

1. 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?)


sonarqube
Look what they need to
mimic a fraction of our power