BEYOND TECHNICAL DEBT

(CodeScene)

LITTLE SURVEY

Do you actually care?

Q&A

- How big is your organisation?
- Do you “measure” your technical debt?
- What kind of legacy code you have?
- Do you know what to refactor?
- Is it easy to prioritise refactorings?
- How do you know that things have actually improved after a refactoring effort?
- Do you have any bottlenecks when multiple developers/teams need to work on the same part of the codebase?
- How do you know which parts of the codebase you should focus on during an off-boarding of a core developer?

WHAT IS TECHNICAL DEBT

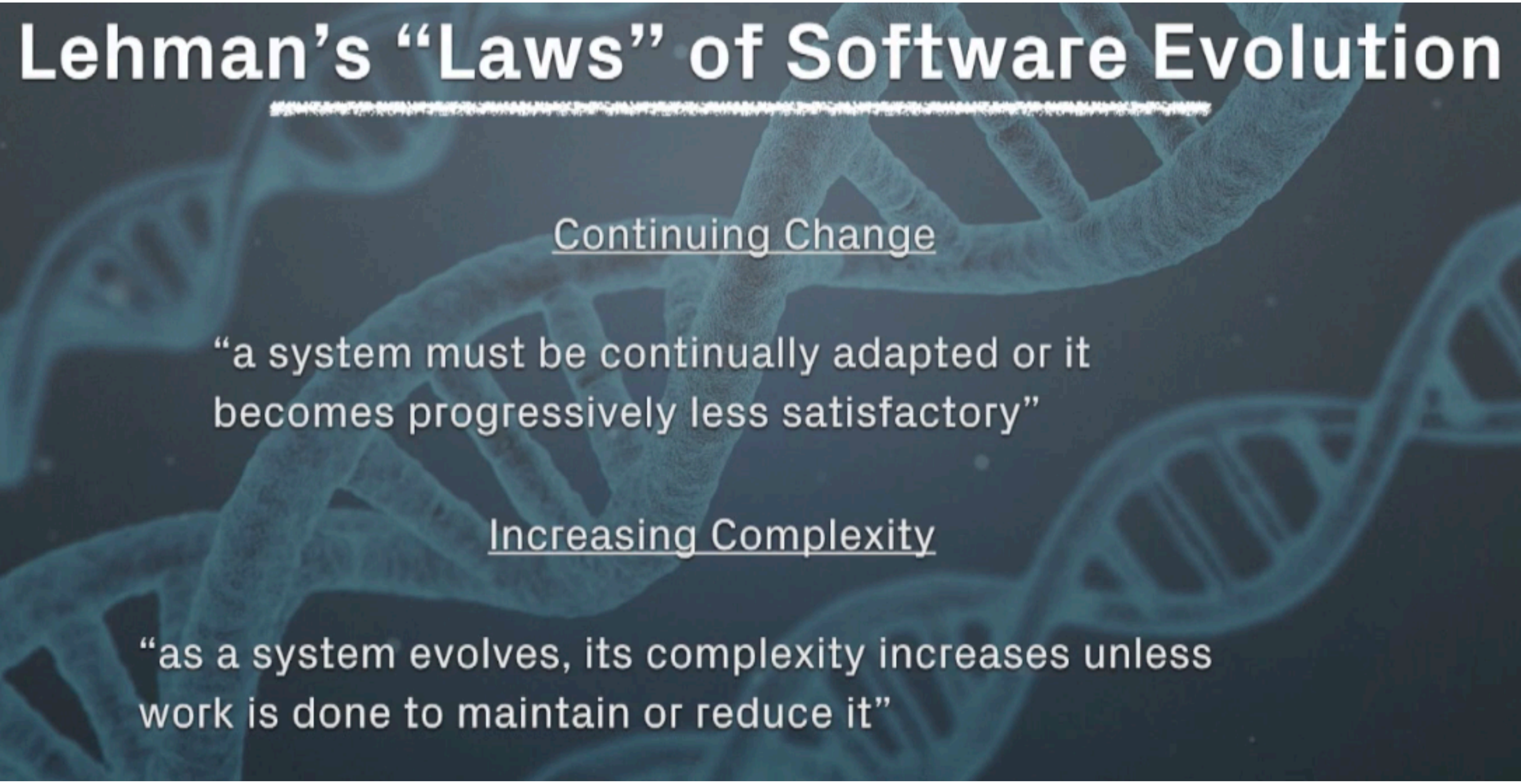
And why you should care…

TECHNICAL DEBT

"Stuff that isn't supposed to be there ***and is in the way*** *of the stuff that is supposed to be there."*

— Building Evolutionary Architectures (p. 110), by Neal Ford, Rebecca Parsons, and Patrick Kia

LEHMAN



Lehman's "Laws" of Software Evolution

Continuing Change

"a system must be continually adapted or it becomes progressively less satisfactory"

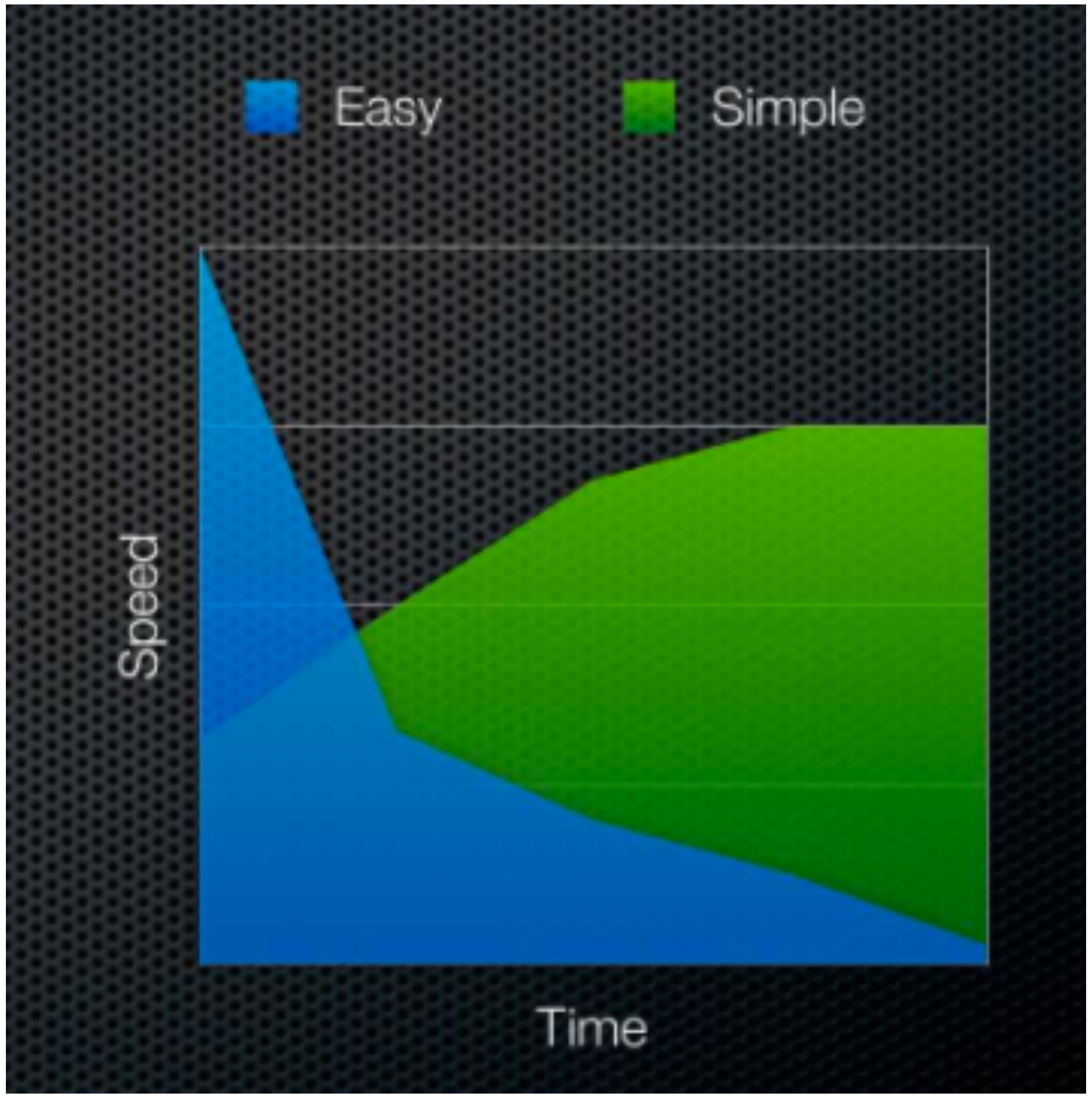
Increasing Complexity

"as a system evolves, its complexity increases unless work is done to maintain or reduce it"

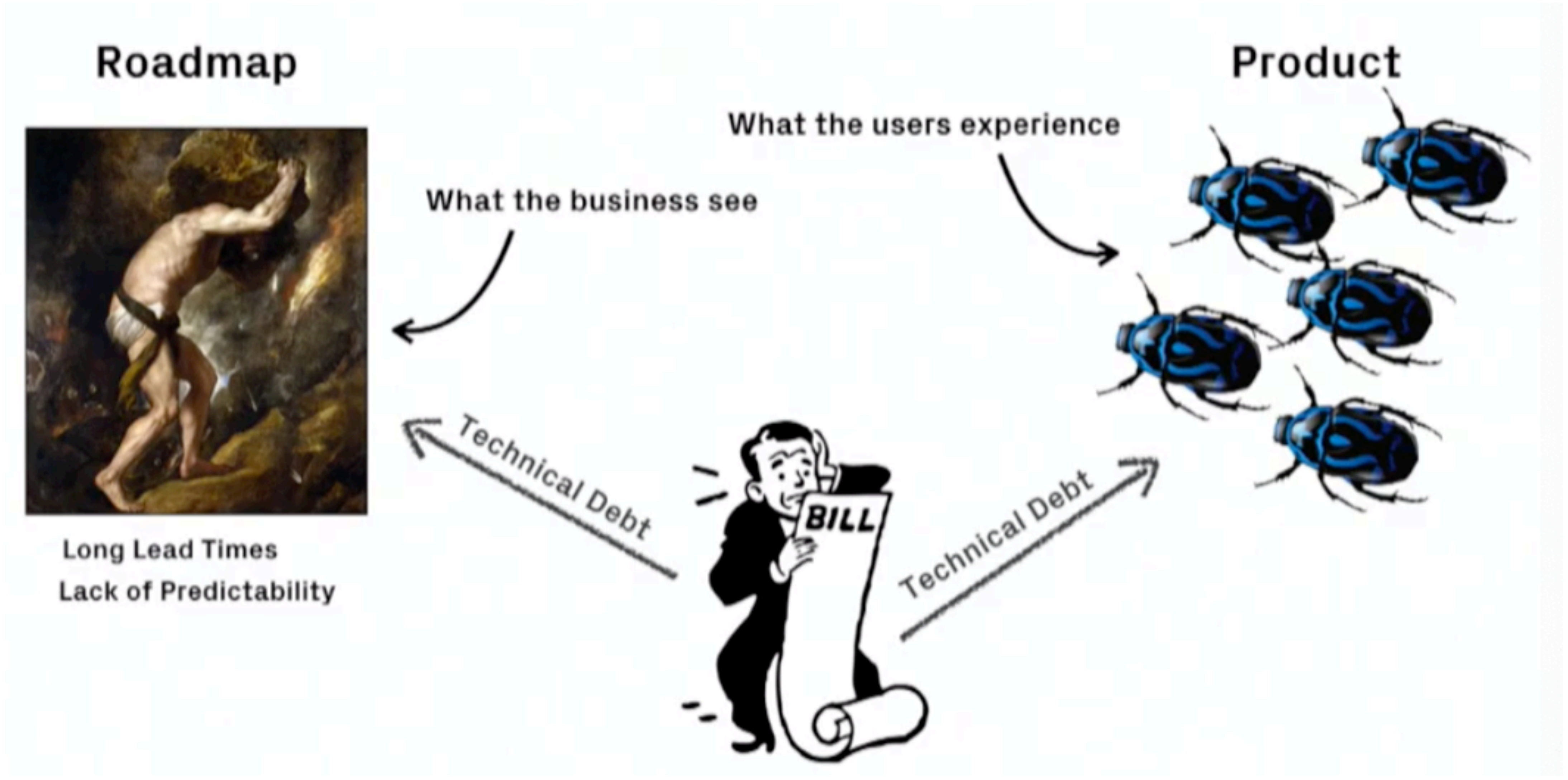
COMPLEXITY KILLS DEVELOPMENT SPEED

"It's my contention, based on experience, that ***if you ignore complexity, you will slow down.*** *You will invariably slow down over the long haul …* ***the complexity will eventually kill you****. It will kill you in a way that will make every sprint accomplish less.*

— Rich Hickey, Simple Made Easy



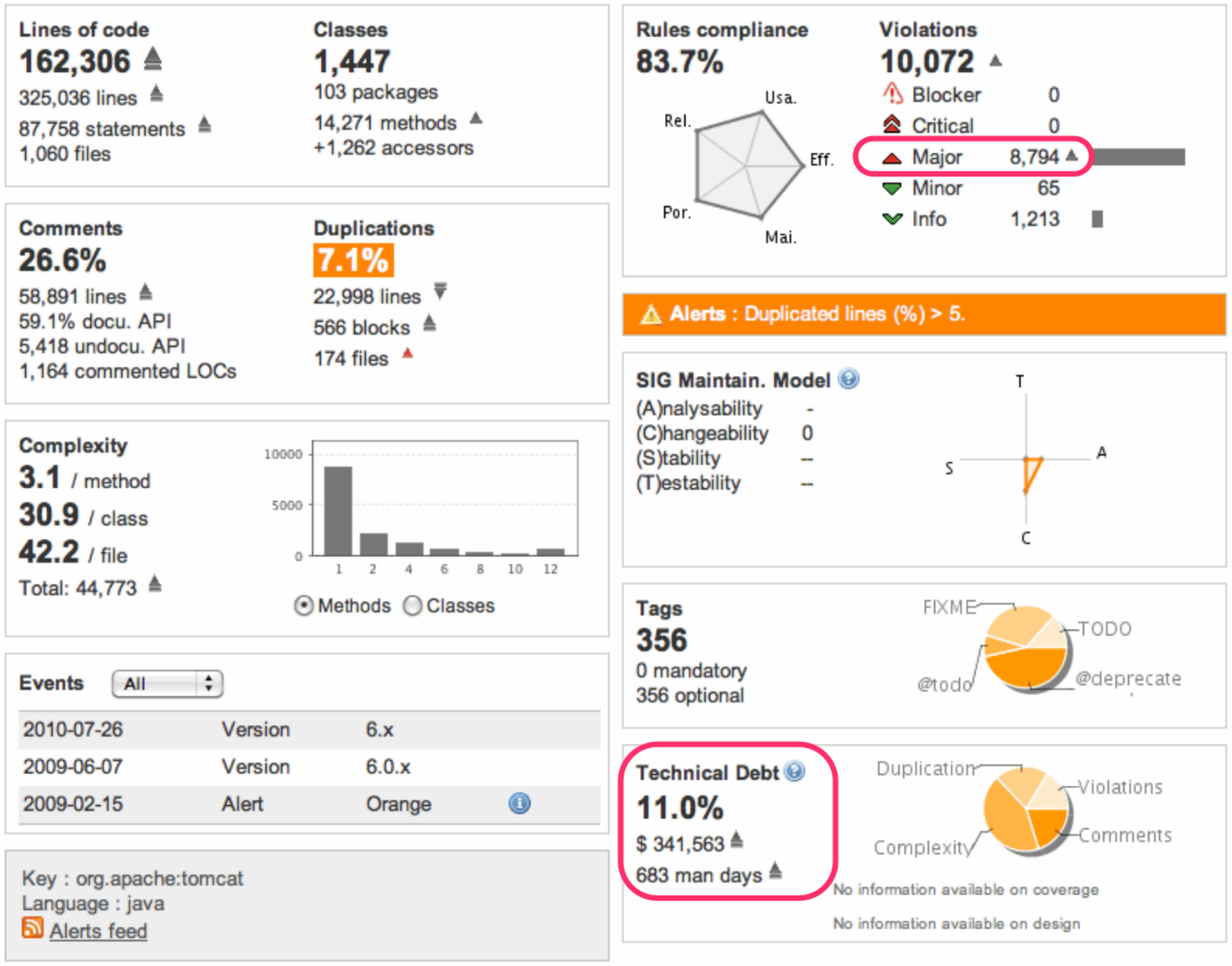
TECHNICAL DEBT HAS IMPACT ON BUSINESS & PRODUCT



CONVENTIONAL TOOLS

What's missing?

ACTIONABLE?



Lines of code
162,306 ▲
325,036 lines ▲
87,758 statements ▲
1,060 files

Classes
1,447
103 packages
14,271 methods ▲
+1,262 accessors

Comments
26.6%
58,891 lines ▲
59.1% docu. API
5,418 undocu. API
1,164 commented LOCs

Duplications
7.1%
22,998 lines ▼
566 blocks ▲
174 files ▲

Complexity
3.1 / method
30.9 / class
42.2 / file
Total: 44,773 ▲

[● Methods] ○ Classes

Events All

2010-07-26	Version	6.x	
2009-06-07	Version	6.0.x	
2009-02-15	Alert	Orange	ⓘ

Key : org.apache:tomcat
Language : java
Alerts feed

Rules compliance
83.7%

Violations
10,072 ▲

Blocker	0
Critical	0
Major	8,794 ▲
Minor	65
Info	1,213

⚠ **Alerts : Duplicated lines (%) > 5.**

SIG Maintain. Model

(A)nalysability	-
(C)hangeability	0
(S)tability	--
(T)estability	--

Tags
356
0 mandatory
356 optional

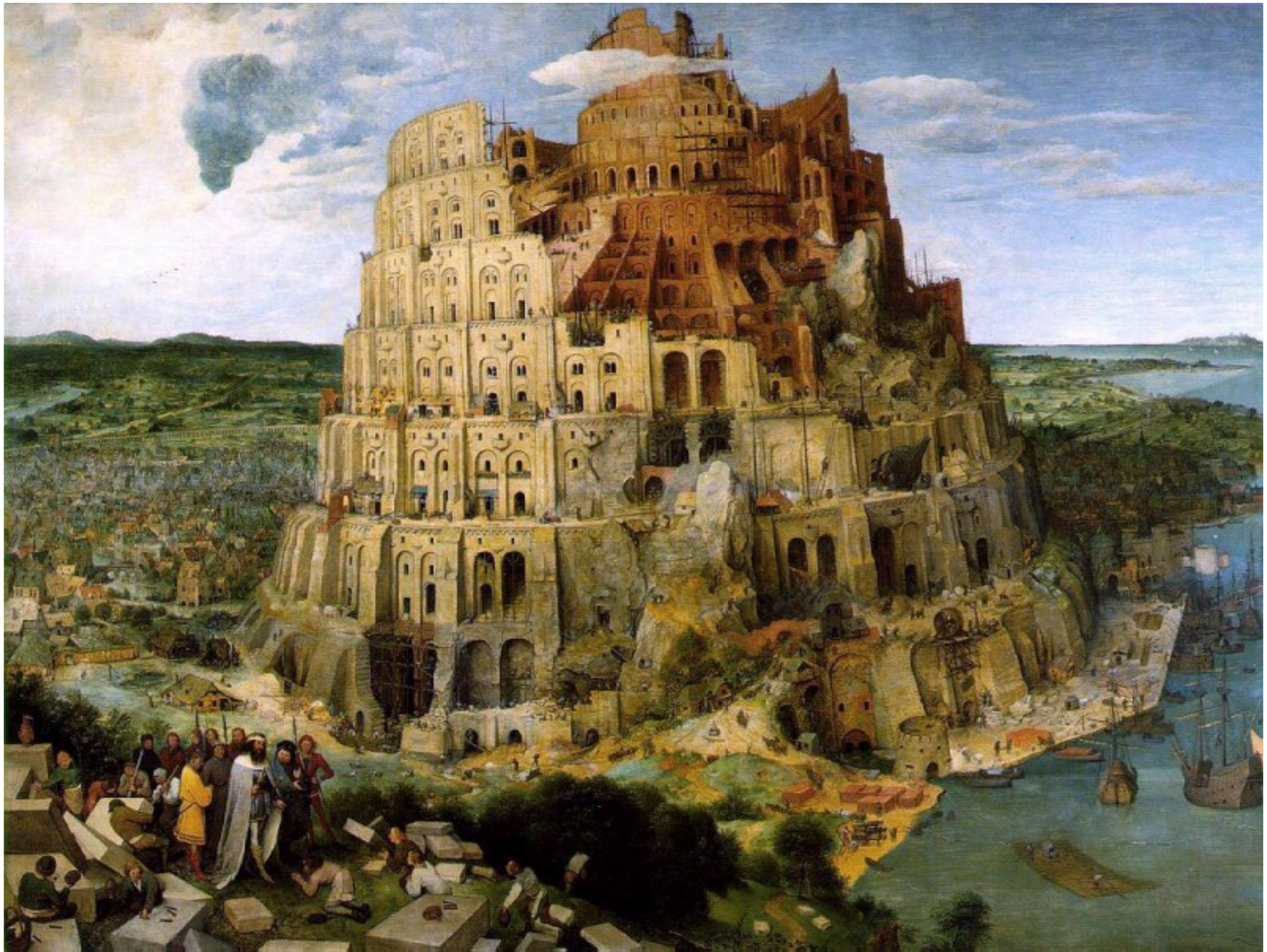
Technical Debt
11.0%
$ 341,563 ▲
683 man days ▲

No information available on coverage
No information available on design

source: http://www.austinjug.org/presentations/HeintzTechnicalDebtSonar.pdf

THOUSANDS OF YEARS OF TECHNICAL DEBT

- Where do you start when you want to pay it back?



CODESCENE

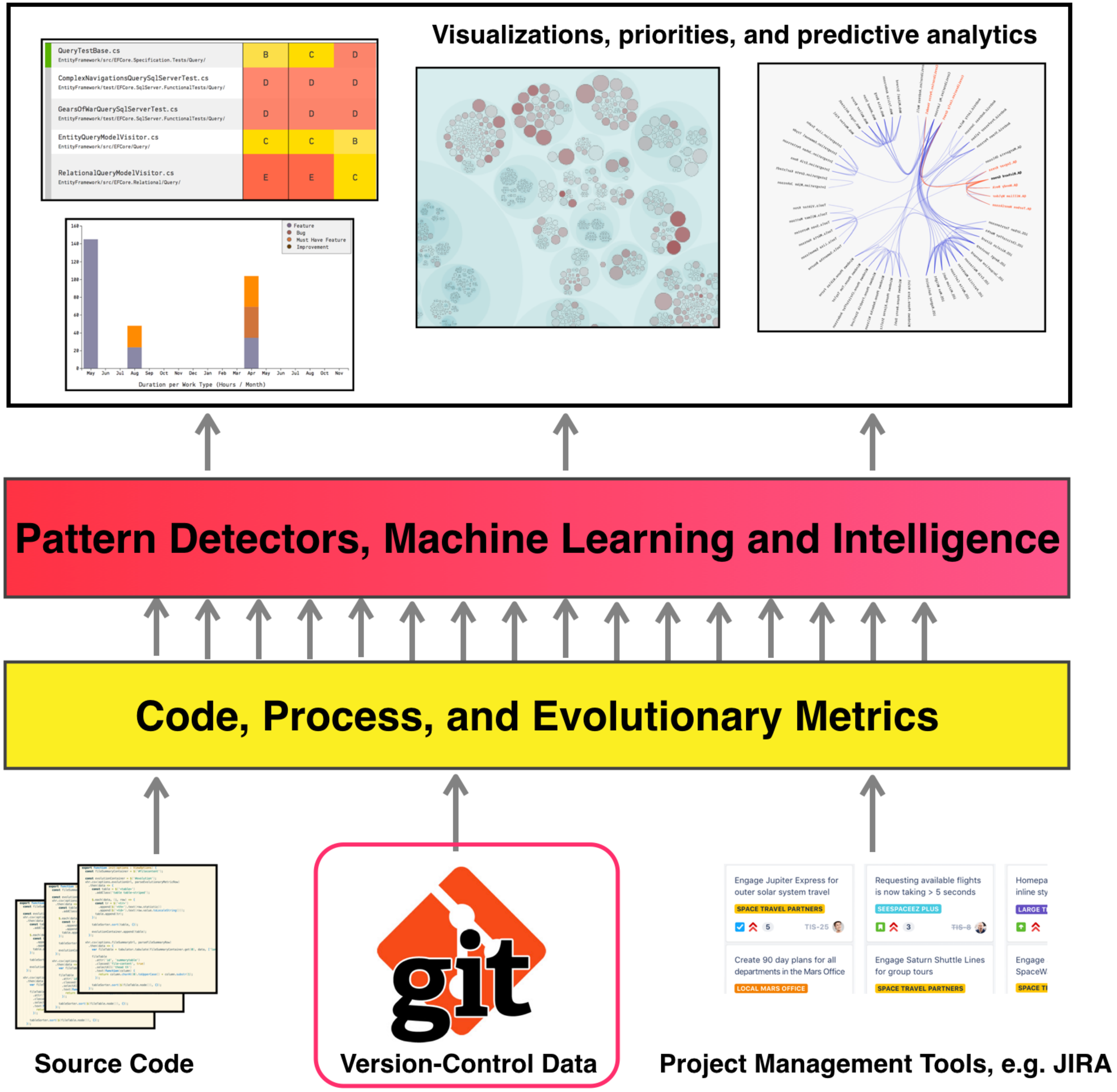
It’s a “movie” rather than a “snapshot”

… static analysis will never be able to tell you if that excess code complexity actually matters – just because a piece of code is complex doesn't mean it's a problem.
CodeScene identifies and prioritizes technical debt based on how the organization works with the code

— How CodeScene Differs From Traditional Code Analysis Tools

+ *Time aspect*

+ *Organization & people*

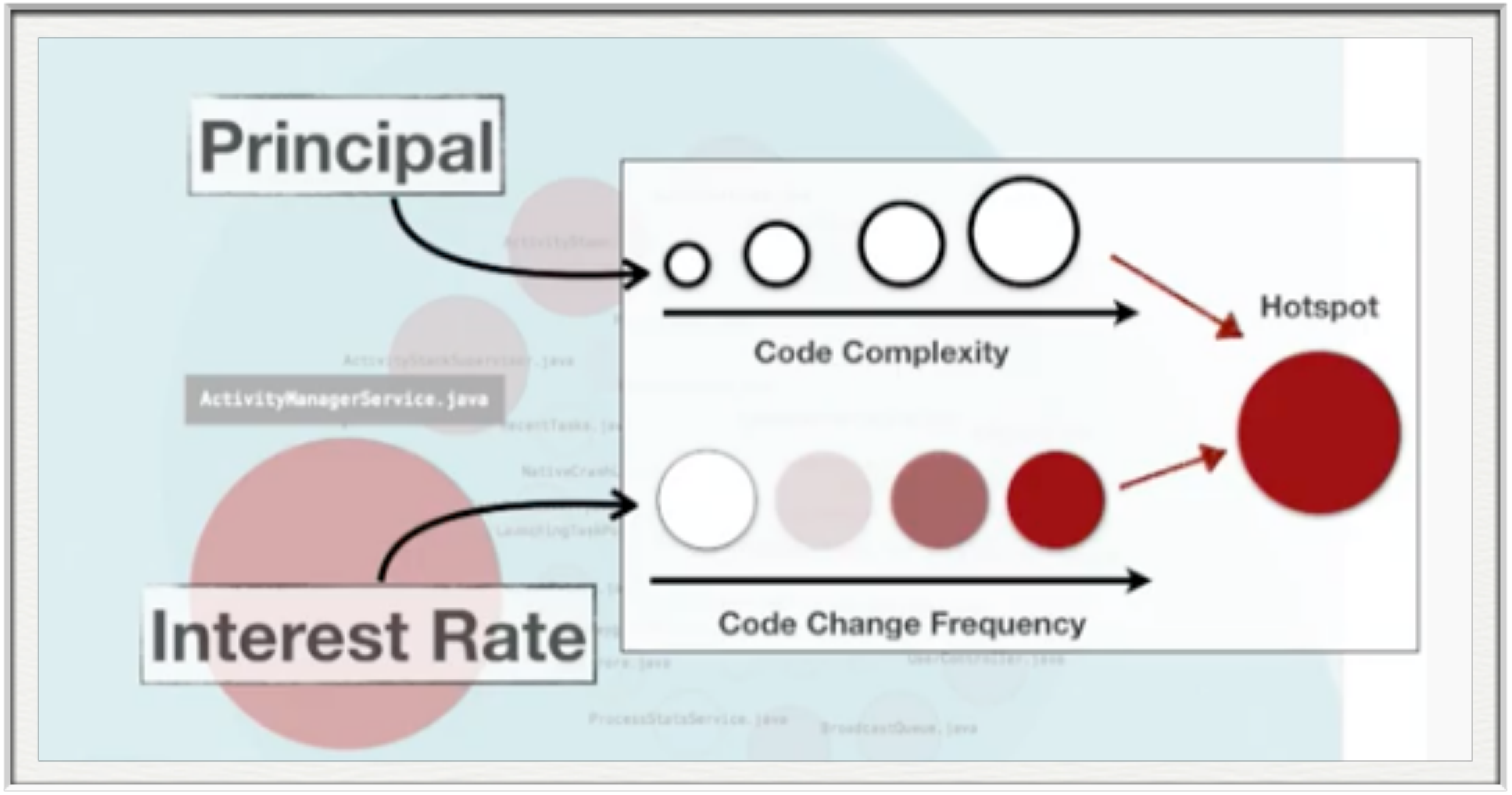

Visualizations, priorities, and predictive analytics
Pattern Detectors, Machine Learning and Intelligence
Code, Process, and Evolutionary Metrics
Source Code
Version-Control Data
Project Management Tools, e.g. JIRA

HOTSPOTS

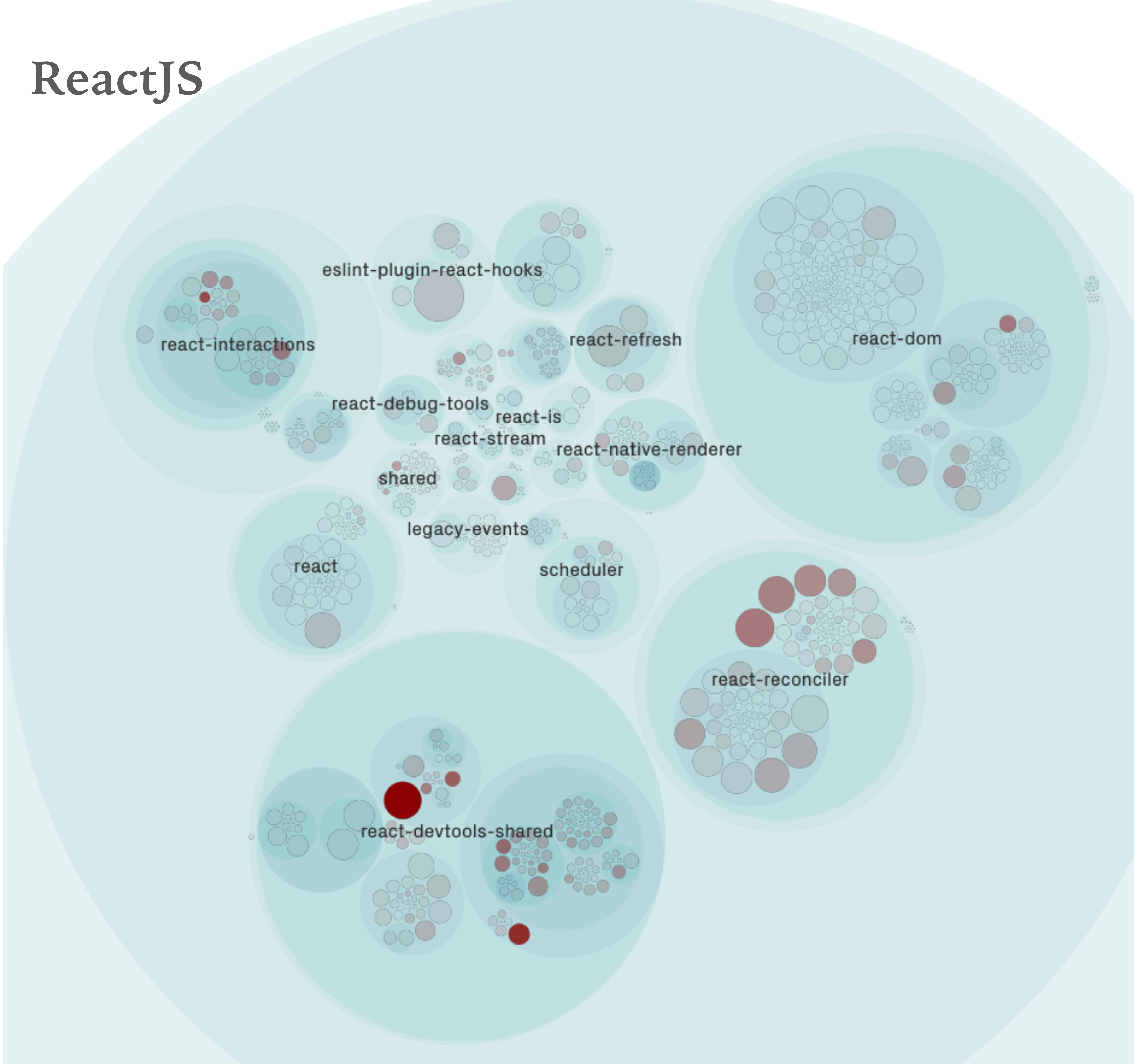
Return On Investment

HOT-WHAT?

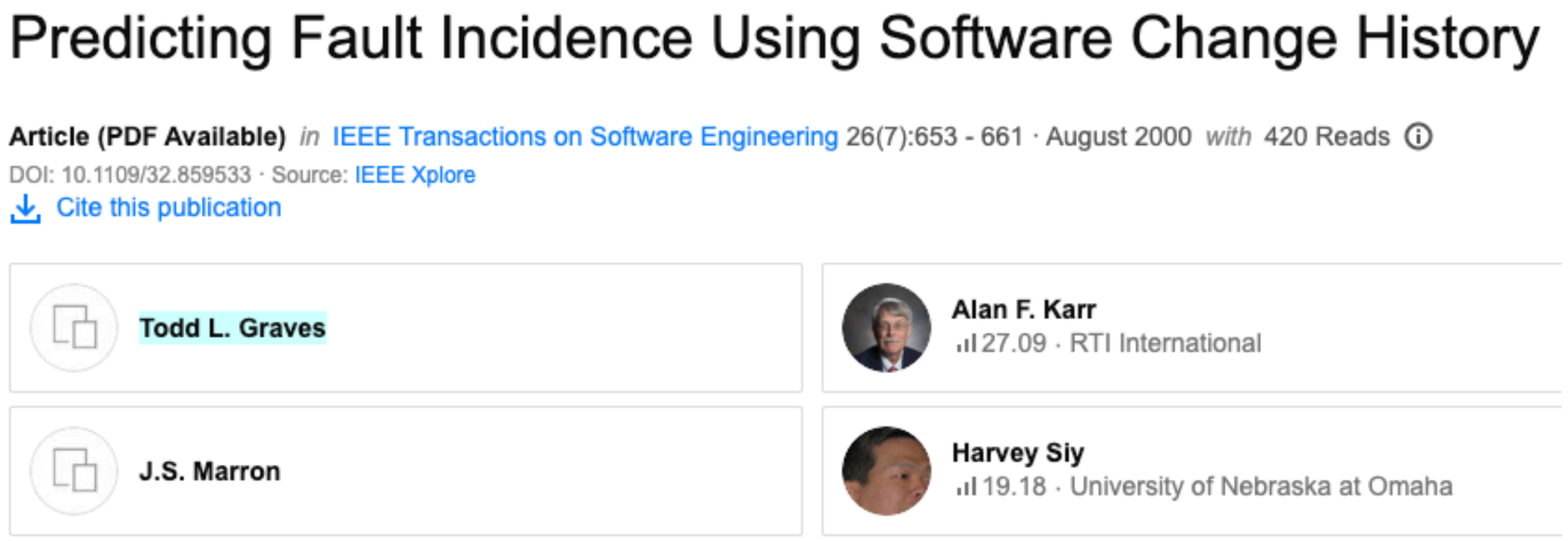
*A hotspot is a **complicated code that you have to work with often.***



ReactJS


eslint-plugin-react-hooks
react-interactions
react-refresh
react-dom
react-debug-tools
react-is
react-stream
react-native-renderer
shared
legacy-events
react
scheduler
react-reconciler
react-devtools-shared

HOTSPOTS ARE ALSO COMMON SOURCES OF BUGS



Predicting Fault Incidence Using Software Change History

Article (PDF Available) *in* IEEE Transactions on Software Engineering 26(7):653 - 661 · August 2000 *with* 420 Reads

DOI: 10.1109/32.859533 · Source: IEEE Xplore

Cite this publication

Todd L. Graves

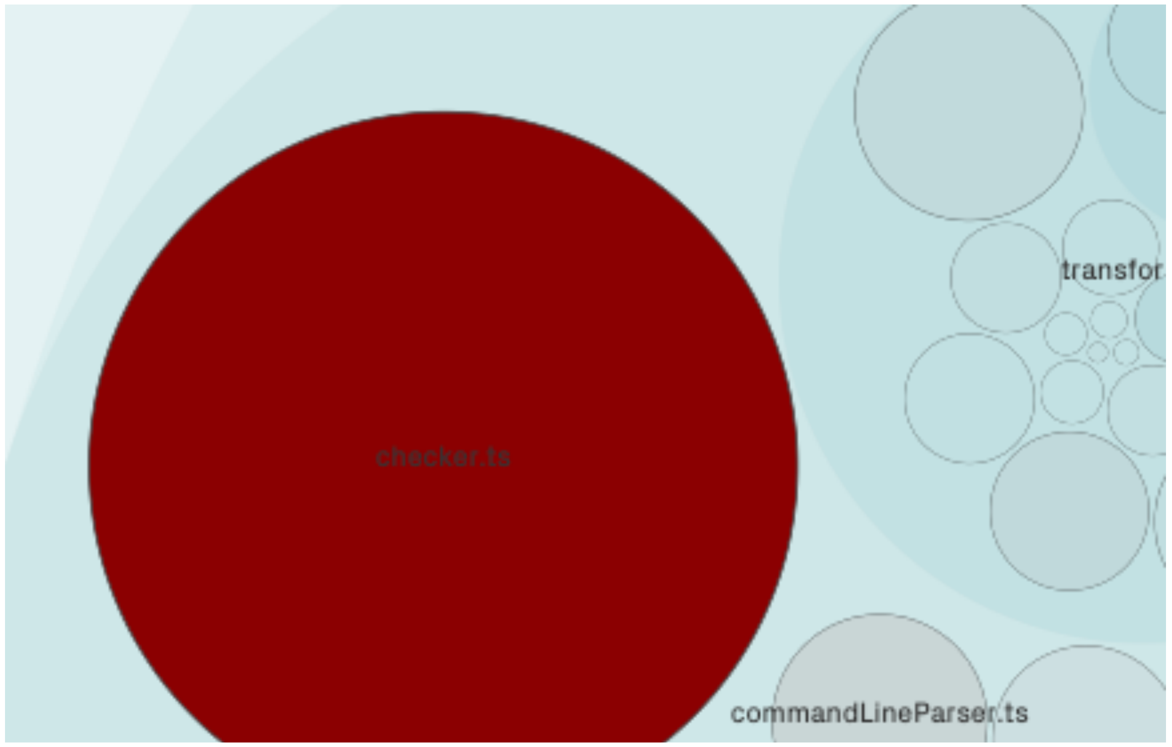
Alan F. Karr
27.09 · RTI International

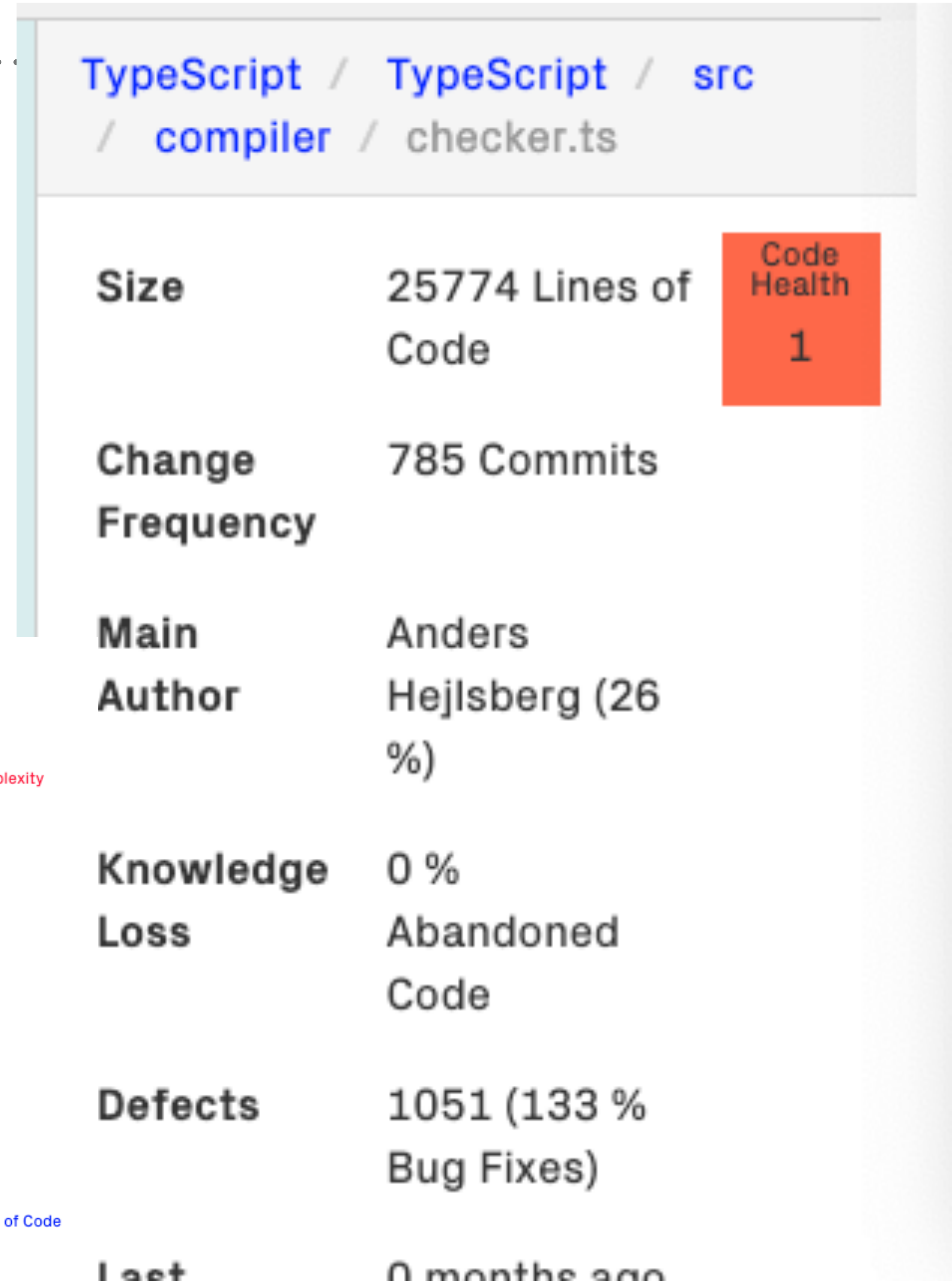
J.S. Marron

Harvey Siy
19.18 · University of Nebraska at Omaha

- *In general, process measures based on the change history are more useful in predicting fault rates than product metrics of the code: For instance, the **number of times code has been changed is a better indication of how many faults it will contain** than is its length.*
- *We also compare the fault rates of code of various ages, finding that if a module is, on the average, a year older than an otherwise similar module, **the older module will have roughly a third fewer faults.** Our most successful model measures the fault potential of a module as the sum of contributions from all of the times the module has been changed, with large, recent changes receiving the most weight*

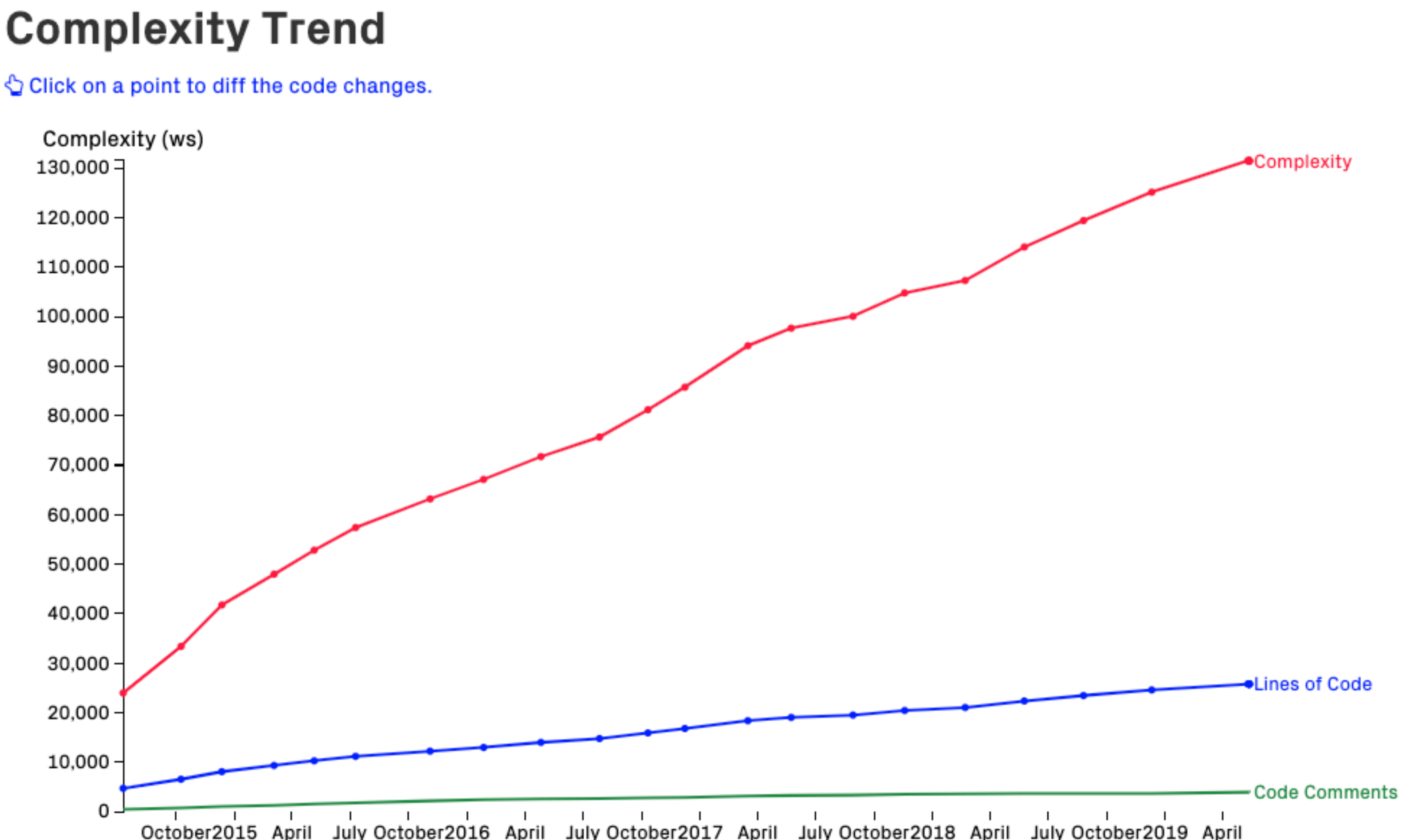
WORRISOME TRENDS





TypeScript / TypeScript / src / compiler / checker.ts

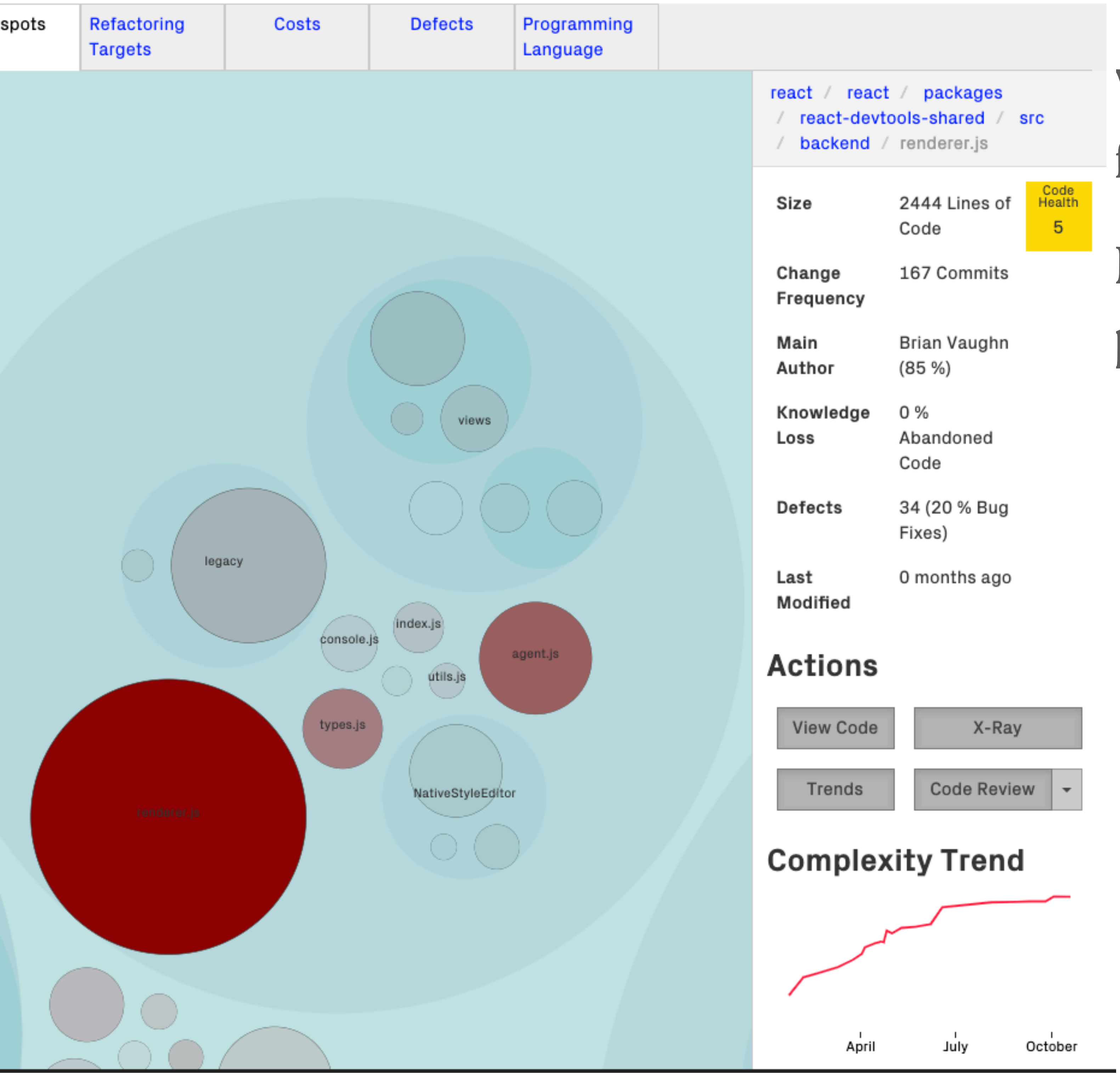
Size	25774 Lines of Code	Code Health 1
Change Frequency	785 Commits	
Main Author	Anders Hejlsberg (26 %)	
Knowledge Loss	0 % Abandoned Code	
Defects	1051 (133 % Bug Fixes)	
Last	0 months ago	



X-RAYS

Deep dive

X-RAY

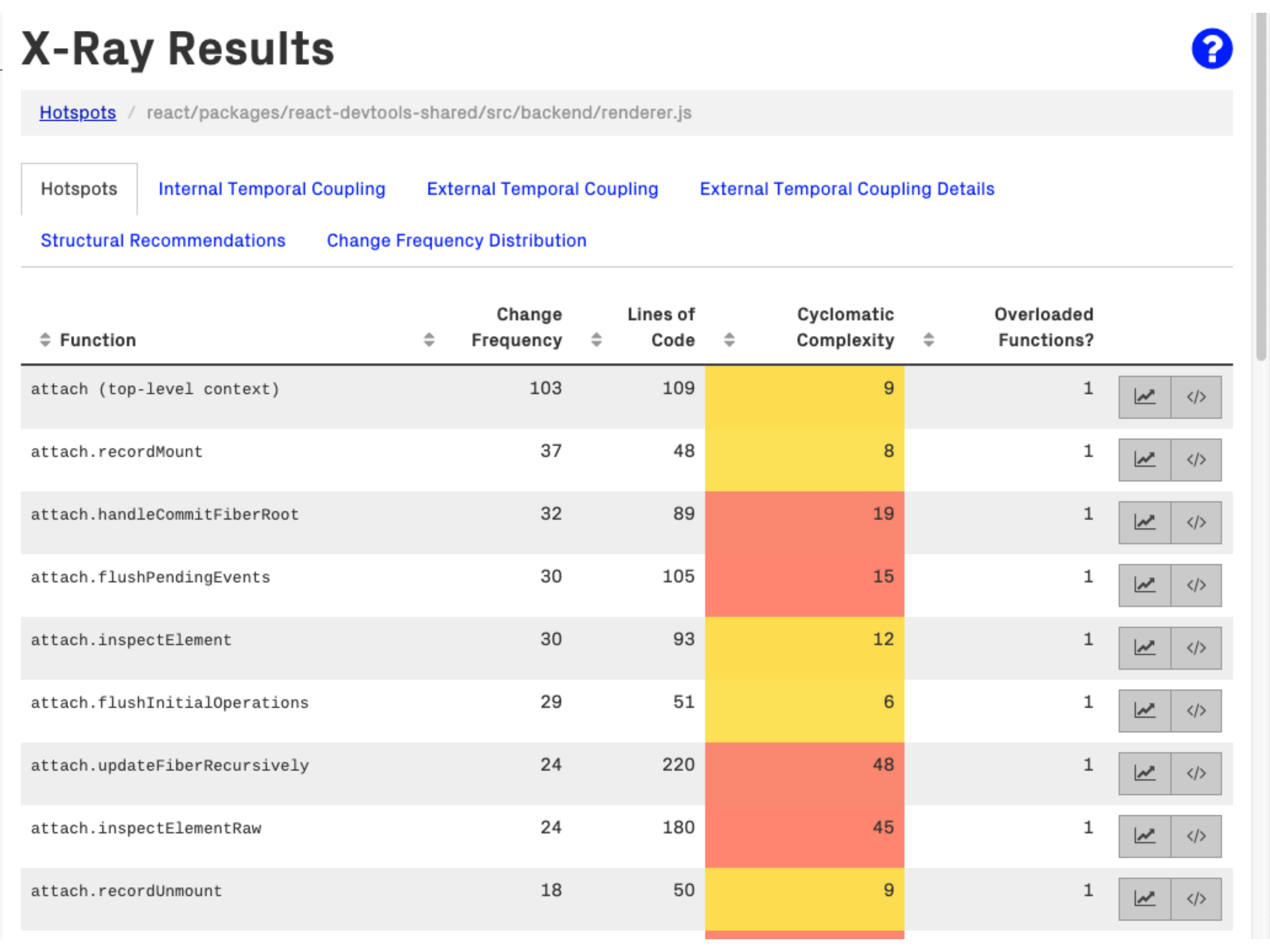


react / react / packages / react-devtools-shared / src / backend / renderer.js

Size	2444 Lines of Code
Change Frequency	167 Commits
Main Author	Brian Vaughn (85 %)
Knowledge Loss	0 % Abandoned Code
Defects	34 (20 % Bug Fixes)
Last Modified	0 months ago

We've identified a problematic file but it's still huge!

Let's *X-Ray* it to find the most problematic functions.



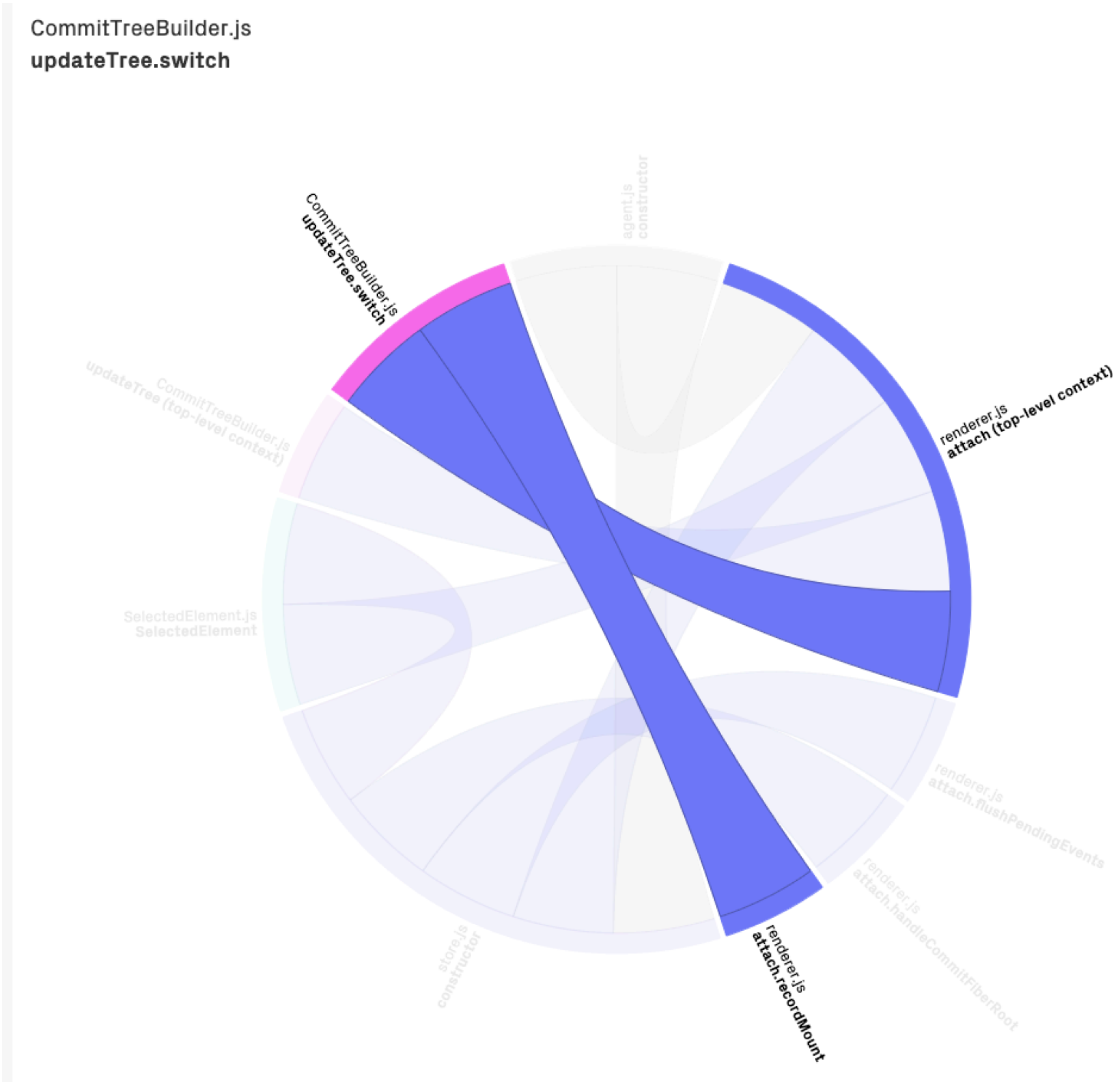
X-Ray Results

Hotspots / react/packages/react-devtools-shared/src/backend/renderer.js

Hotspots | Internal Temporal Coupling | External Temporal Coupling | External Temporal Coupling Details | Structural Recommendations | Change Frequency Distribution

Function	Change Frequency	Lines of Code	Cyclomatic Complexity	Overloaded Functions?
`attach (top-level context)`	103	109	9	1
`attach.recordMount`	37	48	8	1
`attach.handleCommitFiberRoot`	32	89	19	1
`attach.flushPendingEvents`	30	105	15	1
`attach.inspectElement`	30	93	12	1
`attach.flushInitialOperations`	29	51	6	1
`attach.updateFiberRecursively`	24	220	48	1
`attach.inspectElementRaw`	24	180	45	1
`attach.recordUnmount`	18	50	9	1

X-RAY – CHANGE COUPLING

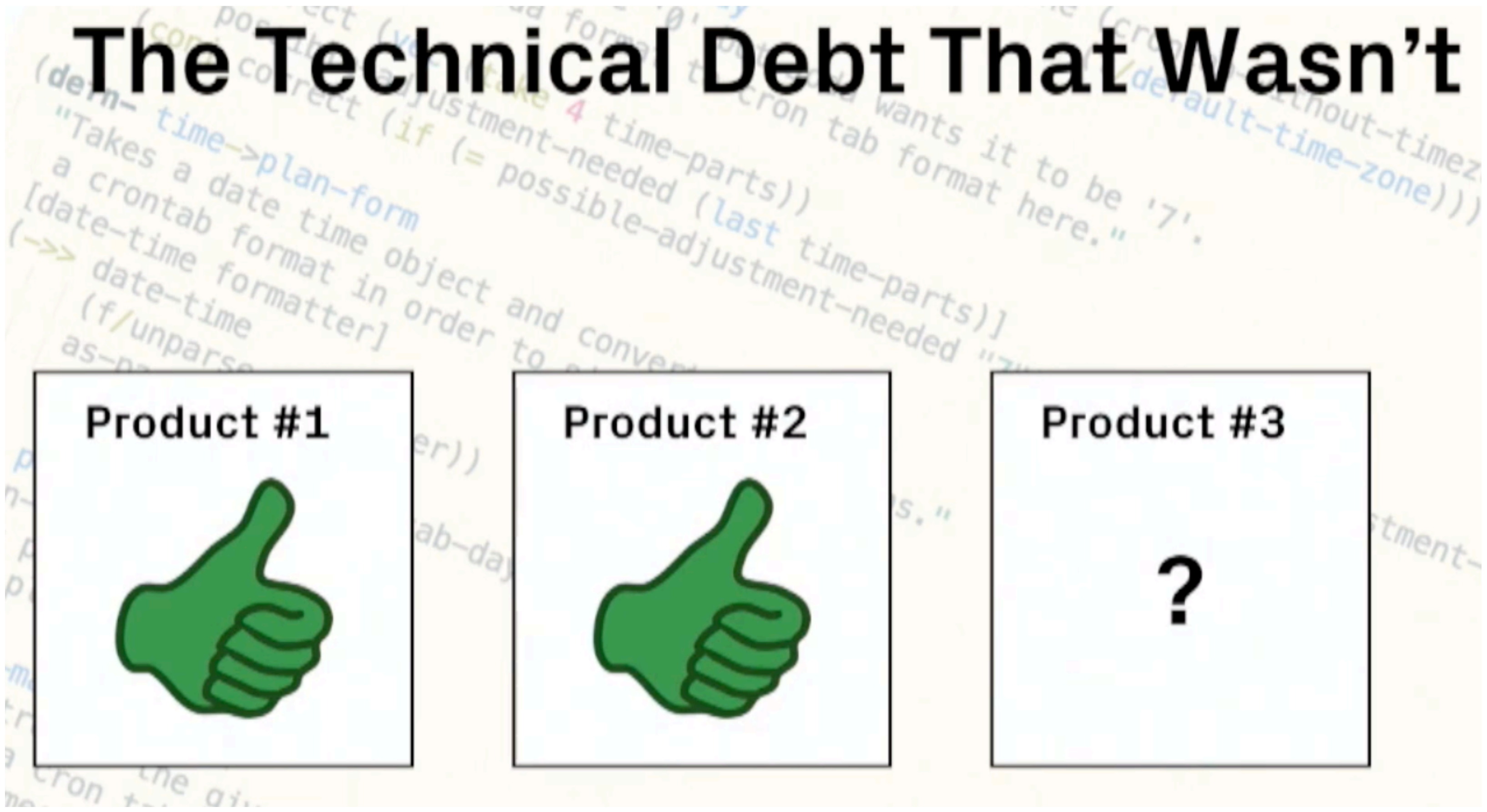

CommitTreeBuilder.js
updateTree.switch
agent.js constructor
CommitTreeBuilder.js updateTree.switch
renderer.js attach (top-level context)
CommitTreeBuilder.js updateTree (top-level context)
SelectedElement.js SelectedElement
renderer.js attach.flushPendingEvents
renderer.js attach.handleCommitFiberRoot
renderer.js attach.recordMount
store.js constructor

LEGACY CODE

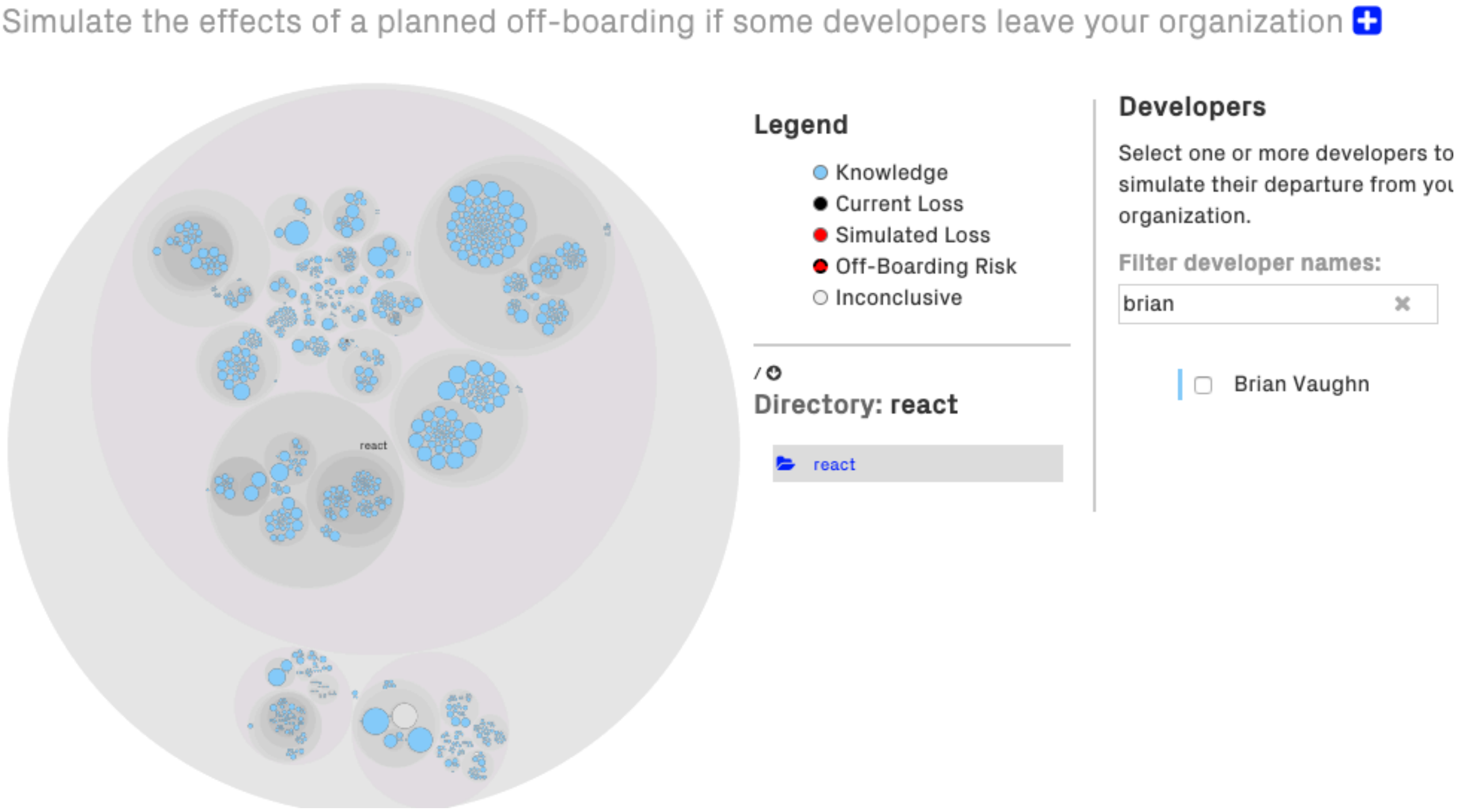
And why technical debt isn't just technical

LEGACY CODE

- Legacy code is typically used to describe the code that:
  - lacks in quality (relative perspective)
  - **we didn't write ourselves**



HOW QUICKLY CAN YOU TURN YOUR CODEBASE INTO LEGACY CODE?



Simulate the effects of a planned off-boarding if some developers leave your organization ➕

Legend

- ● Knowledge
- ● Current Loss
- ● Simulated Loss
- ● Off-Boarding Risk
- ○ Inconclusive

/

Directory: react

react

Developers

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

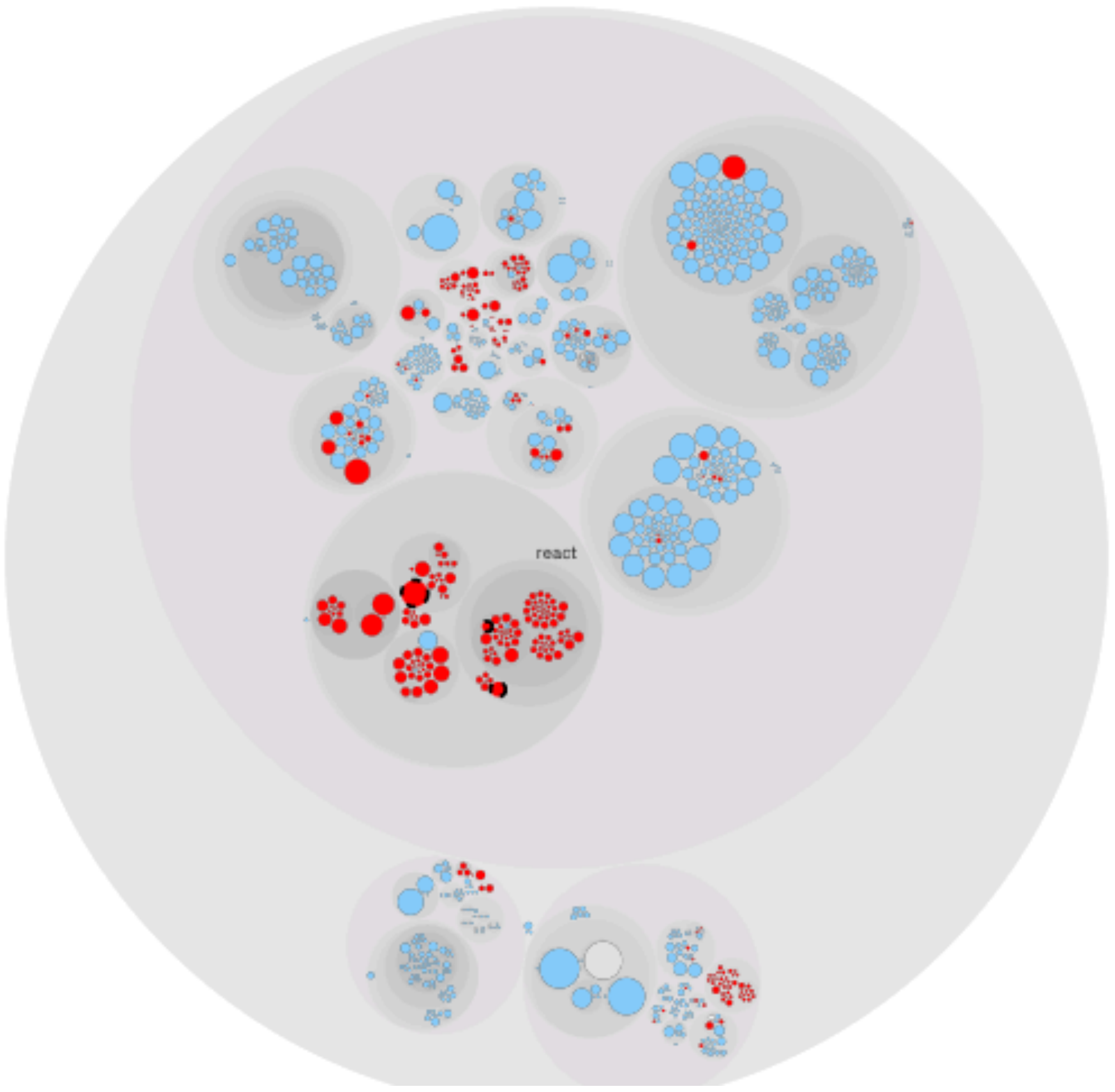
Filter developer names:

brian ✖

☐ Brian Vaughn

AFTER THEY LEAVE …

Simulate the effects of a planned off-boarding if some developers leave your organization ➕



Legend

- ● Knowledge
- ● Current Loss
- ● Simulated Loss
- ● Off-Boarding Risk
- ○ Inconclusive

/

Directory: react

react

Simulated offboarded authors

☑ Brian Vaughn

Developers

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

Filter developer names:

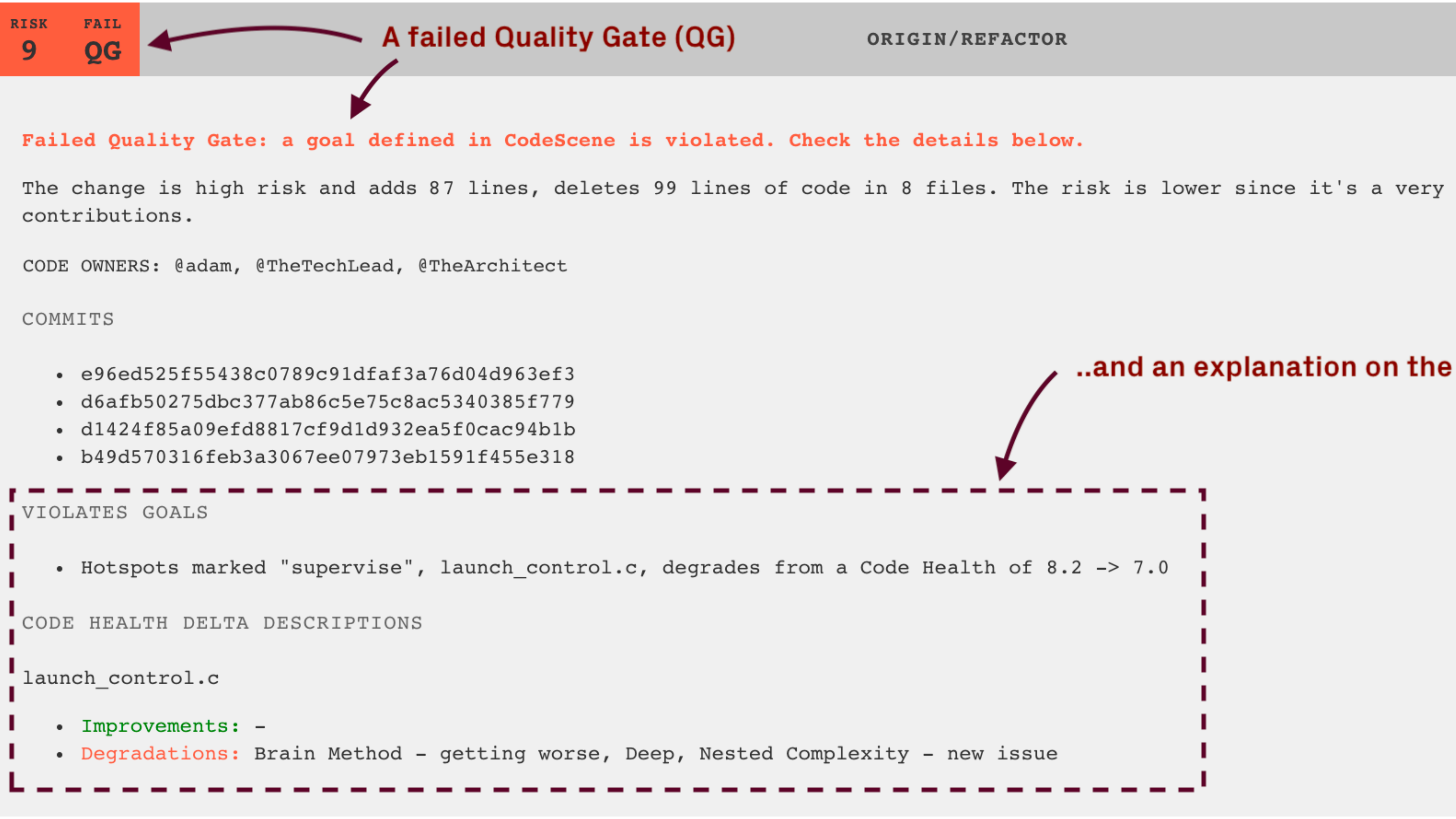
brian ✖

☑ Brian Vaughn

ON CODE REVIEWS

… and what to focus on

CODESCENE DELTA ANALYSIS (JENKINS)



Failed Quality Gate: a goal defined in CodeScene is violated. Check the details below.

The change is high risk and adds 87 lines, deletes 99 lines of code in 8 files. The risk is lower since it's a very contributions.

CODE OWNERS: @adam, @TheTechLead, @TheArchitect

COMMITS

- e96ed525f55438c0789c91dfaf3a76d04d963ef3
- d6afb50275dbc377ab86c5e75c8ac5340385f779
- d1424f85a09efd8817cf9d1d932ea5f0cac94b1b
- b49d570316feb3a3067ee07973eb1591f455e318

VIOLATES GOALS

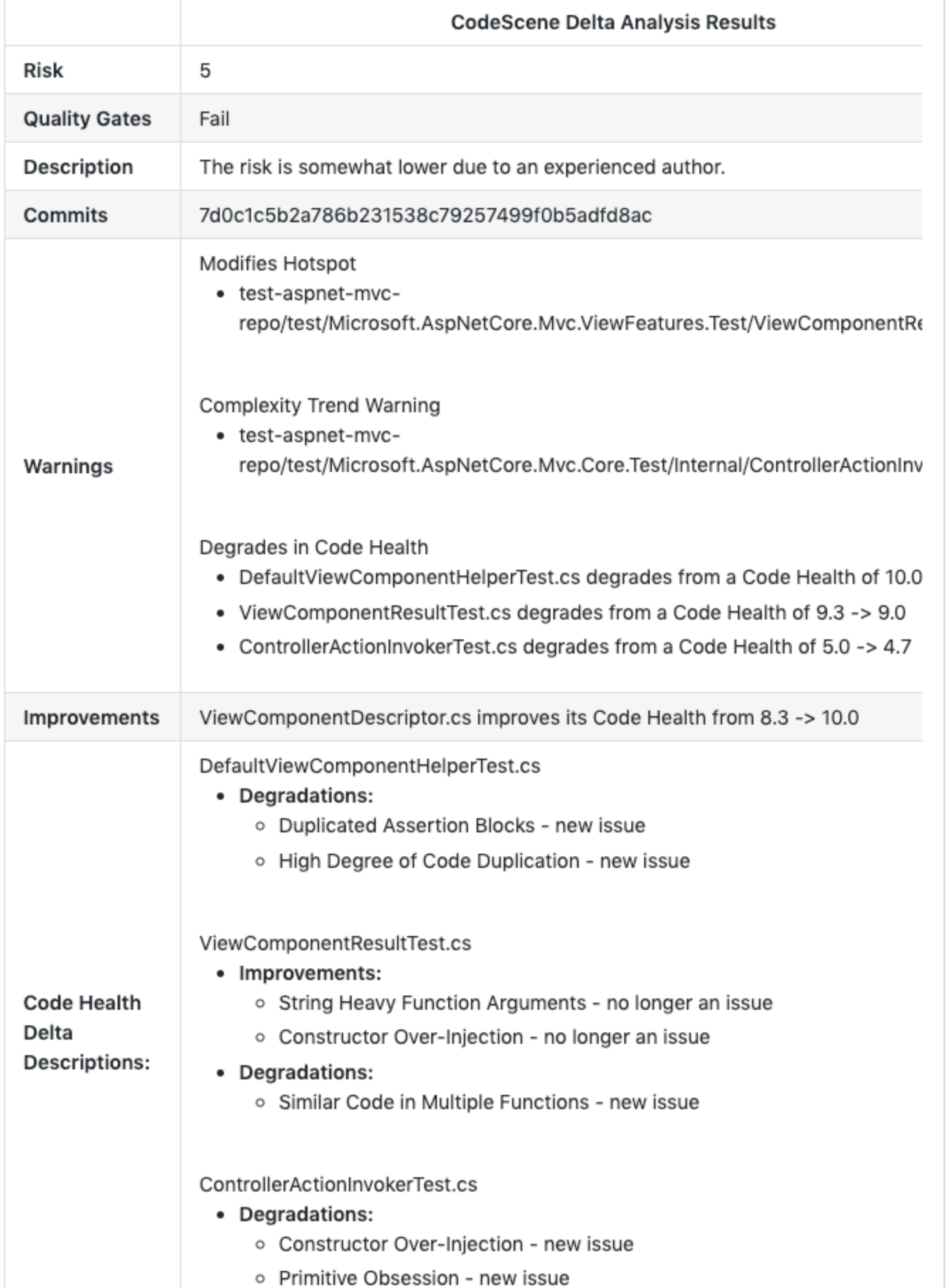
- Hotspots marked "supervise", launch_control.c, degrades from a Code Health of 8.2 -> 7.0

CODE HEALTH DELTA DESCRIPTIONS

launch_control.c

- Improvements: -
- Degradations: Brain Method - getting worse, Deep, Nested Complexity - new issue

PR COMMENTS



	CodeScene Delta Analysis Results
Risk	5
Quality Gates	Fail
Description	The risk is somewhat lower due to an experienced author.
Commits	7d0c1c5b2a786b231538c79257499f0b5adfd8ac
Warnings	Modifies Hotspot • test-aspnet-mvc-repo/test/Microsoft.AspNetCore.Mvc.ViewFeatures.Test/ViewComponentRe Complexity Trend Warning • test-aspnet-mvc-repo/test/Microsoft.AspNetCore.Mvc.Core.Test/Internal/ControllerActionInv Degrades in Code Health • DefaultViewComponentHelperTest.cs degrades from a Code Health of 10.0 • ViewComponentResultTest.cs degrades from a Code Health of 9.3 -> 9.0 • ControllerActionInvokerTest.cs degrades from a Code Health of 5.0 -> 4.7
Improvements	ViewComponentDescriptor.cs improves its Code Health from 8.3 -> 10.0
Code Health Delta Descriptions:	DefaultViewComponentHelperTest.cs • **Degradations:** ◦ Duplicated Assertion Blocks - new issue ◦ High Degree of Code Duplication - new issue ViewComponentResultTest.cs • **Improvements:** ◦ String Heavy Function Arguments - no longer an issue ◦ Constructor Over-Injection - no longer an issue • **Degradations:** ◦ Similar Code in Multiple Functions - new issue ControllerActionInvokerTest.cs • **Degradations:** ◦ Constructor Over-Injection - new issue ◦ Primitive Obsession - new issue

AND MORE

Always keep exploring

THERE'S MUCH MORE IN CODESCENE

- Change coupling
- Microservices
  - Shotgun surgery
  - Team conflicts
  - Technical sprawl
- Proactive warnings
- Retrospectives
- Delivery Performance
- Branch Analyses

TO CONCLUDE…

- Technical debt is a real problem regardless of programming language
- There's a huge amount of useful information stored in your version control system
- Ultimately, you need to rely on human expertise
- Support your developer's judgment and experience with data to get the highest ROI

RESOURCES

Where to learn more

- codescene.io
- codescene.io/showcase (React, ASP.NET, Rails, …)
- CodeScene blog: https://www.empear.com/blog/
- How CodeScene Differs From Traditional Code Analysis Tools
- Adam Tornhill - Talk Session: Prioritizing Technical Debt as if Time and Money Matters
- Software Design X-Rays (the book)
- Predicting Fault Incidence Using Software Change History
- Simple Made Easy
- Elements of Clojure

???

APPENDINX

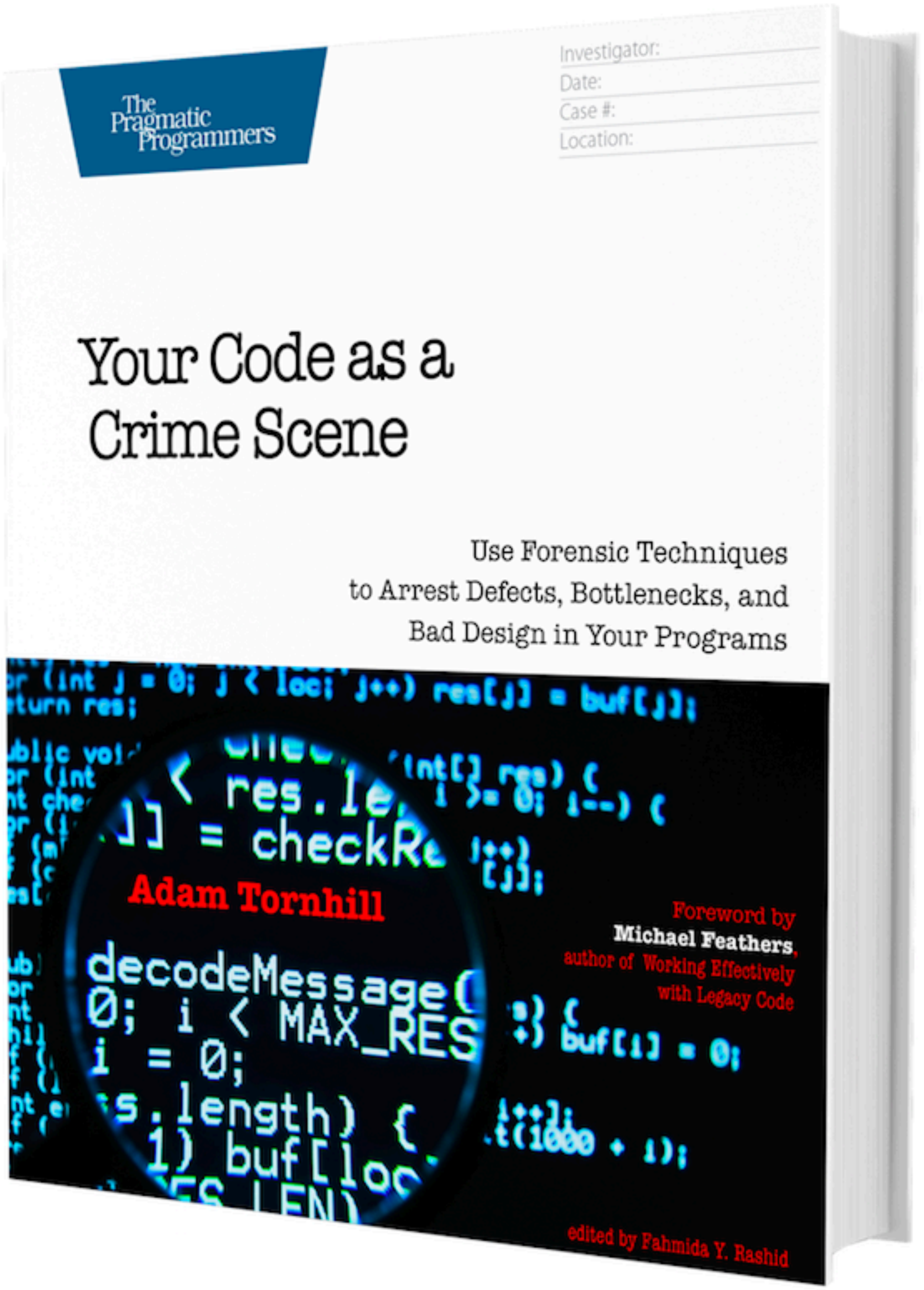
Keep Fun

ALAN PERLIS

***I think that it's extraordinarily important that we in computer science keep fun in computing**. When it started out, it was an awful lot of fun. Of course, the paying customers got shafted every now and then, and after a while we began to take their complaints seriously. **We began to feel as if we really were responsible for the successful, error-free perfect use of these machines. I don't think we are**. I think we're responsible for stretching them, setting them off in new directions, and keeping fun in the house. I hope the field of computer science never loses its sense of fun. **Above all, I hope we don't become missionaries**. Don't feel as if you're Bible salesmen. The world has too many of those already. What you know about computing other people will learn. Don't feel as if the key to successful computing is only in your hands. What's in your hands, I think and hope, is **intelligence**: the ability to see the machine as more than when you were first led up to it, that you can make it more.*

- Quoted in *The Structure and Interpretation of Computer Programs* by Hal Abelson, Gerald Jay Sussman and Julie Sussman (McGraw-Hill, 2nd edition, 1996)

WHY CODE-SCENE?


The Pragmatic Programmers
Investigator:
Date:
Case #:
Location:
Your Code as a Crime Scene
Use Forensic Techniques to Arrest Defects, Bottlenecks, and Bad Design in Your Programs
Adam Tornhill
Foreword by Michael Feathers, author of Working Effectively with Legacy Code
edited by Fahmida Y. Rashid