yonder

R&D Metrics, Reloaded

Portfolio Managers Focus Group

2025 Oct, UK

Confidential — Not for public distribution

agenda

1 **Introduction**

What is technical risk and technical debt

2 **Measuring**

How is software quality measured

3 **Impact**

How do we calculate the impact?

Remus Pereni

CTO / Software Architect

Technical University of Cluj Napoca

BS, Computer Science

- **1996 – Nethrom (Yonder)**
- **2000 – Startup**
- **2005 – Yonder / SD, PM, DM, Architect**
- **2017 – CTO**

2016 start of the Technology DDs

200+ Technology DD Reports

TSS Blue, TSS Public, Vela, Harris, Perseus, CSI, Strikwerda Investments, Jonas, Volaris

The impact of code and technical debt

1 Introduction

Technical Risk

The potential for losses due to failures or shortcomings in technology systems, processes, or implementations that can impact project outcomes or business objectives.

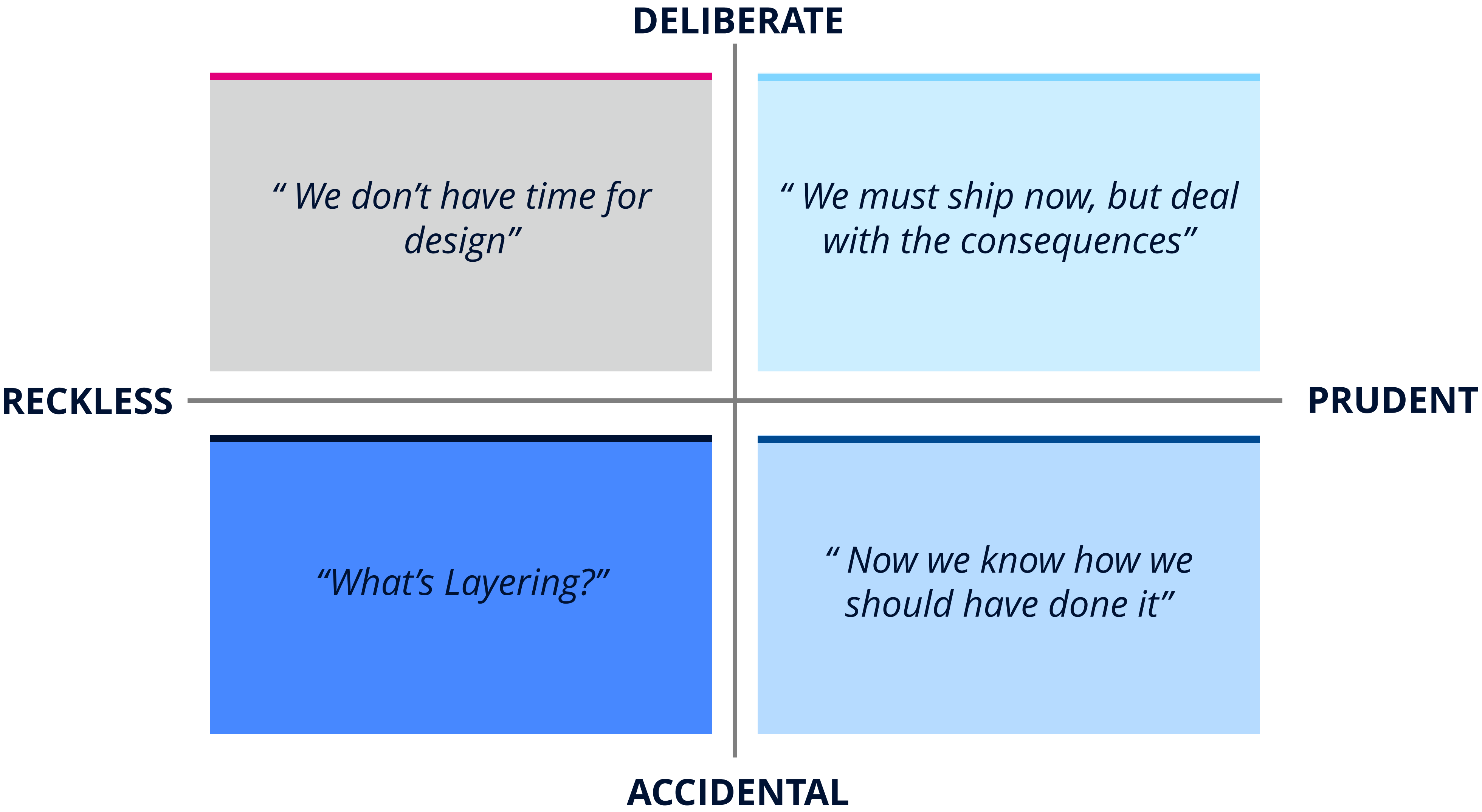
Is that different from **Technical Debt**?

Technical Debt

The accumulated costs and future liabilities resulting from shortcuts or suboptimal technical decisions made during the software development process. It represents the work that needs to be done before a piece of software can be considered complete or optimal.

- Code debt
- Architectural debt
- Technology debt
- Testing debt
- Infrastructural debt
- Know-how debt

Technical Debt Quadrants



DELIBERATE

RECKLESS

PRUDENT

ACCIDENTAL

" We don't have time for design"

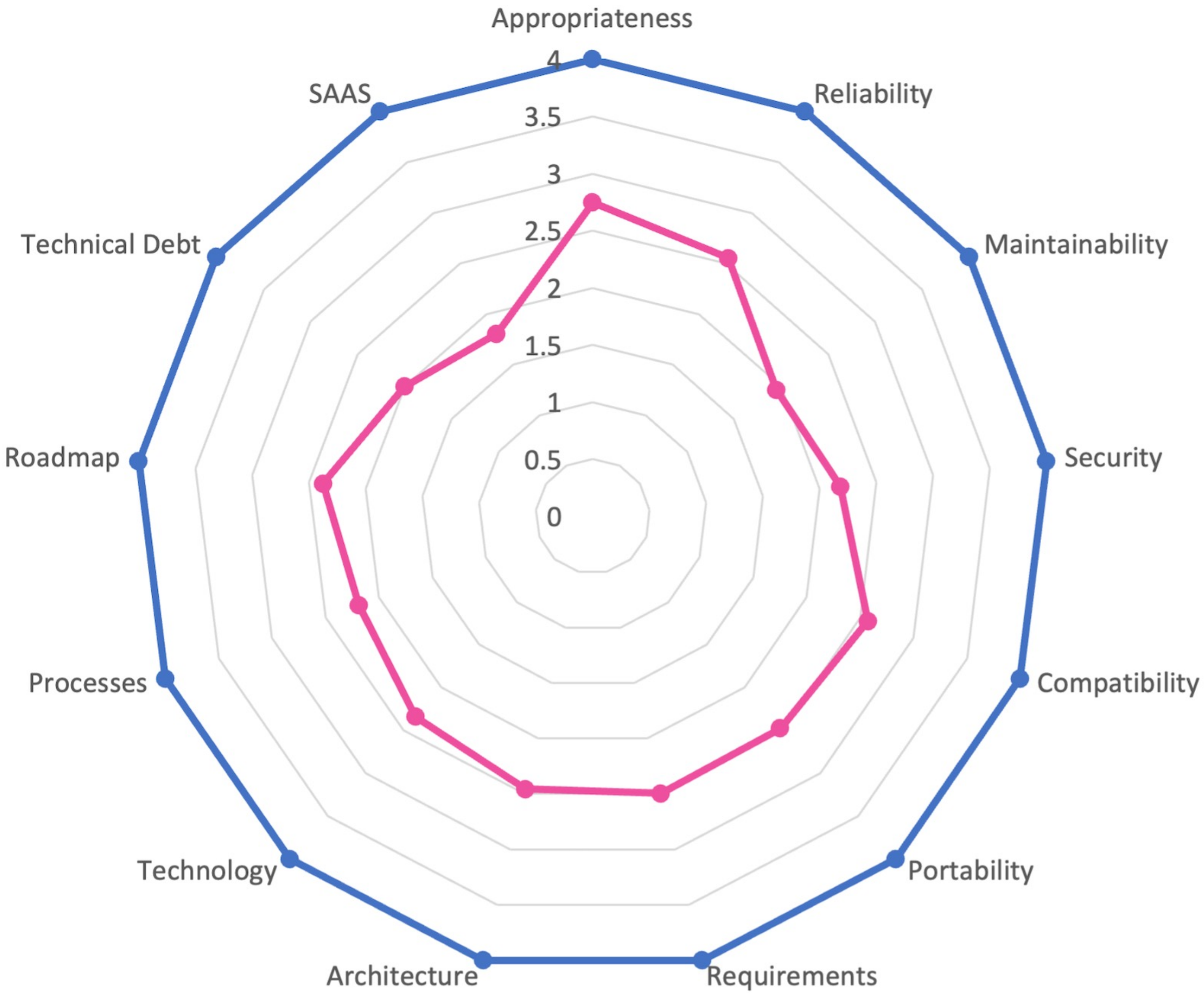
" We must ship now, but deal with the consequences"

"What's Layering?"

" Now we know how we should have done it"

Does your portfolio have a technical debt problem?

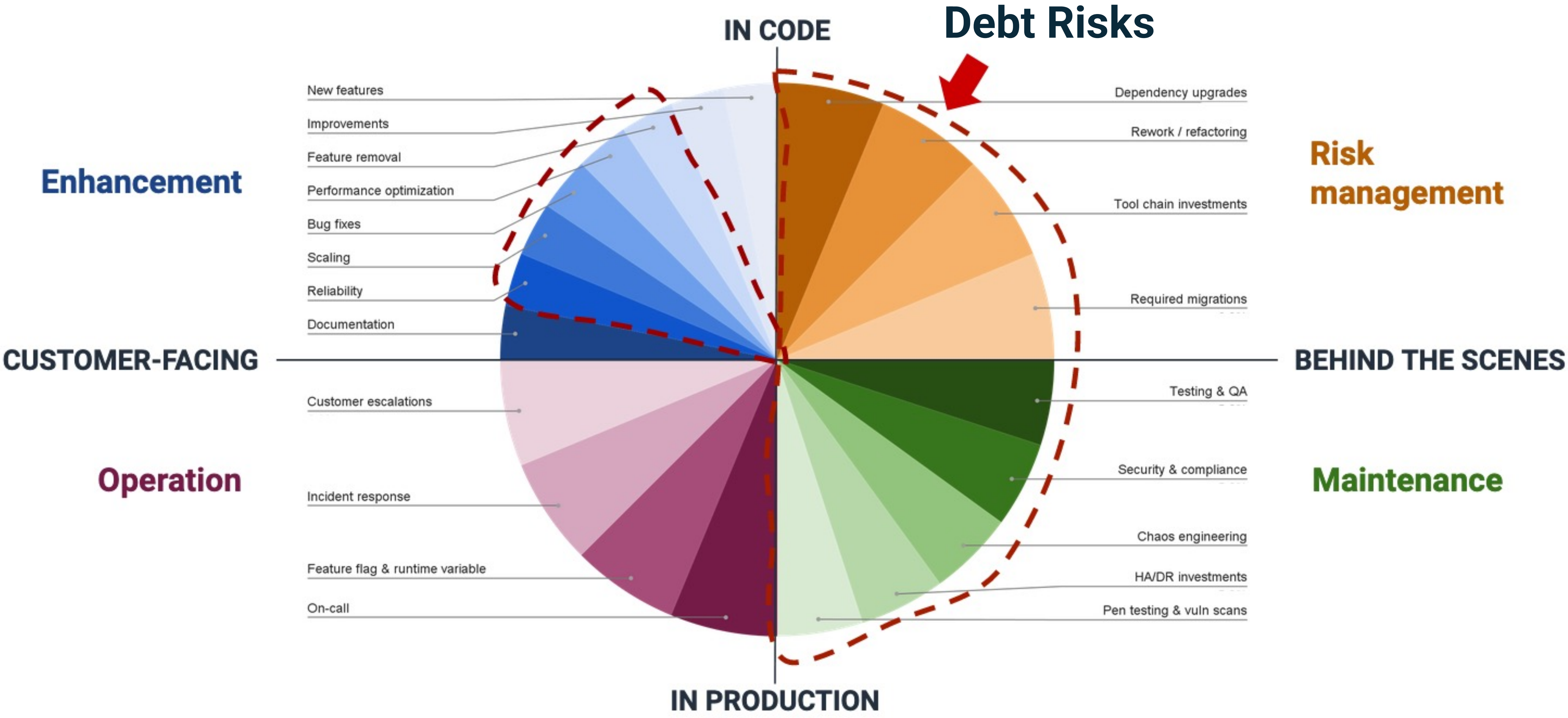
Do we see a tech debt problem?



Avg of 144 products seen in the past years on M&A

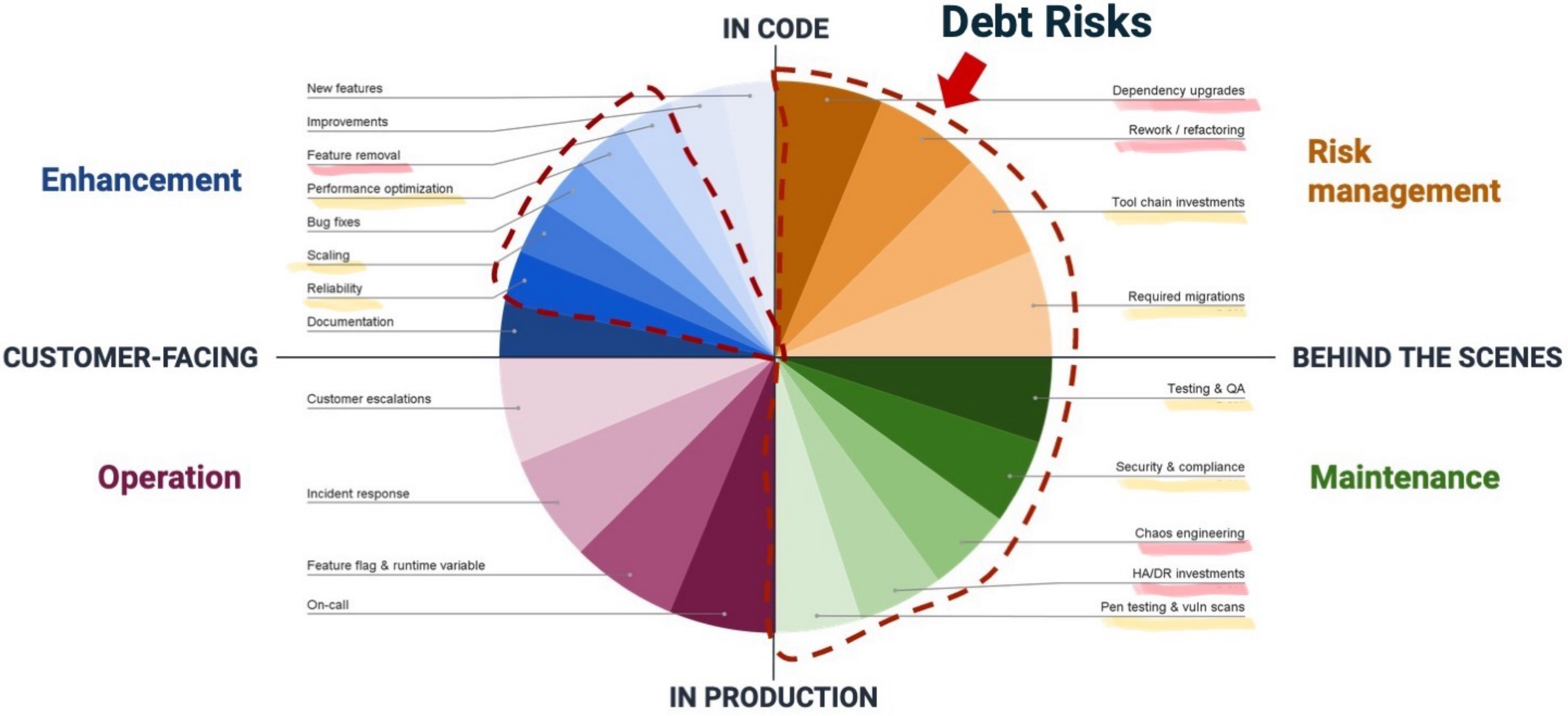
yonder

Technical Debt Quadrants



Is there **focus** for **risk mitigation**? **Which items**?

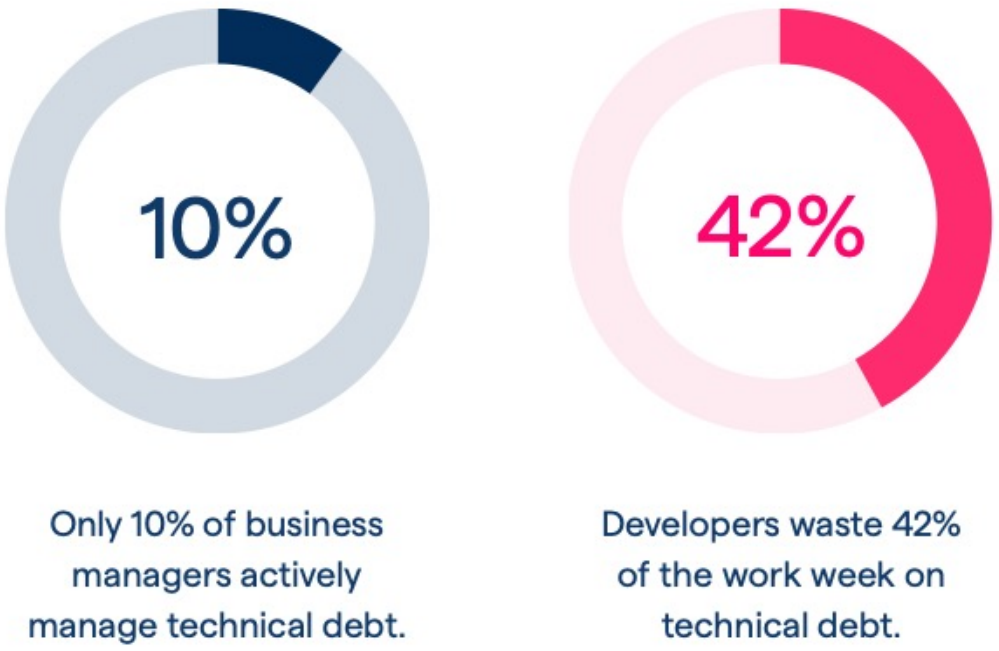
Technical Debt Quadrants



What is your process to manage technical debt in your portfolio?

yonder

Who measures technical debt?



Only 10% of business managers actively manage technical debt.

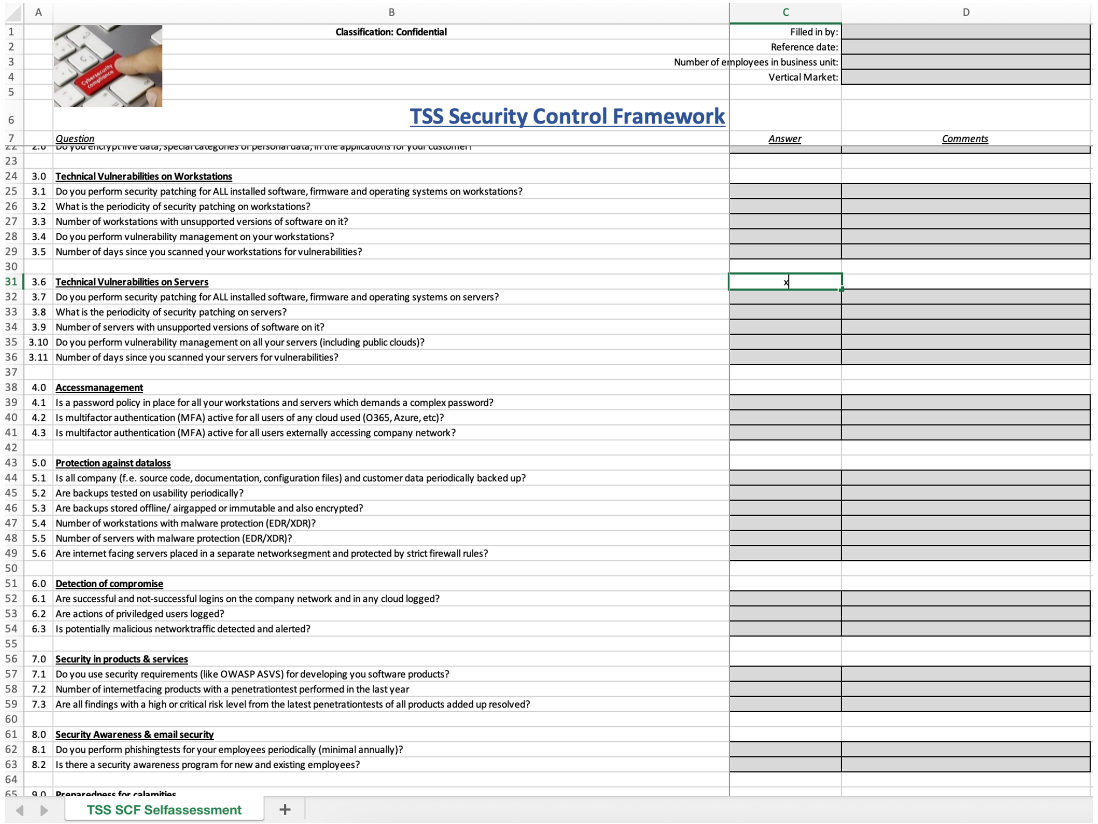
Developers waste 42% of the work week on technical debt.

none of the interviewed companies had a clear strategy on how to track and address the wasted time

Besker, T., Martini, A., Bosch, J. (2019) “Software Developer Productivity Loss Due to Technical Debt”

Antonio Martini, Terese Besker, and Jan Bosch. 2018. Technical debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations. Science of Computer Programming 163 (2018), 42–61.

TSS Security Control Framework a.k.a. TSS SCF



	A	B	C	D
1		Classification: Confidential	Filled in by:	
2			Reference date:	
3			Number of employees in business unit:	
4			Vertical Market:	
5				
6		TSS Security Control Framework		
7		*Question*	*Answer*	*Comments*
22	2.6	Do you encrypt live data, special categories of personal data, in the applications for your customer?		
23				
24	3.0	**Technical Vulnerabilities on Workstations**		
25	3.1	Do you perform security patching for ALL installed software, firmware and operating systems on workstations?		
26	3.2	What is the periodicity of security patching on workstations?		
27	3.3	Number of workstations with unsupported versions of software on it?		
28	3.4	Do you perform vulnerability management on your workstations?		
29	3.5	Number of days since you scanned your workstations for vulnerabilities?		
30				
31	3.6	**Technical Vulnerabilities on Servers**	x	
32	3.7	Do you perform security patching for ALL installed software, firmware and operating systems on servers?		
33	3.8	What is the periodicity of security patching on servers?		
34	3.9	Number of servers with unsupported versions of software on it?		
35	3.10	Do you perform vulnerability management on all your servers (including public clouds)?		
36	3.11	Number of days since you scanned your servers for vulnerabilities?		
37				
38	4.0	**Accessmanagement**		
39	4.1	Is a password policy in place for all your workstations and servers which demands a complex password?		
40	4.2	Is multifactor authentication (MFA) active for all users of any cloud used (O365, Azure, etc)?		
41	4.3	Is multifactor authentication (MFA) active for all users externally accessing company network?		
42				
43	5.0	**Protection against dataloss**		
44	5.1	Is all company (f.e. source code, documentation, configuration files) and customer data periodically backed up?		
45	5.2	Are backups tested on usability periodically?		
46	5.3	Are backups stored offline/ airgapped or immutable and also encrypted?		
47	5.4	Number of workstations with malware protection (EDR/XDR)?		
48	5.5	Number of servers with malware protection (EDR/XDR)?		
49	5.6	Are internet facing servers placed in a separate networksegment and protected by strict firewall rules?		
50				
51	6.0	**Detection of compromise**		
52	6.1	Are successful and not-successful logins on the company network and in any cloud logged?		
53	6.2	Are actions of priviledged users logged?		
54	6.3	Is potentially malicious networktraffic detected and alerted?		
55				
56	7.0	**Security in products & services**		
57	7.1	Do you use security requirements (like OWASP ASVS) for developing you software products?		
58	7.2	Number of internetfacing products with a penetrationtest performed in the last year		
59	7.3	Are all findings with a high or critical risk level from the latest penetrationtests of all products added up resolved?		
60				
61	8.0	**Security Awareness & email security**		
62	8.1	Do you perform phishingtests for your employees periodically (minimal annually)?		
63	8.2	Is there a security awareness program for new and existing employees?		
64				
65	9.0	**Preparedness for calamities**		

In TSS the TSS SCF was introduced in 2022 and covers areas that are related to security:

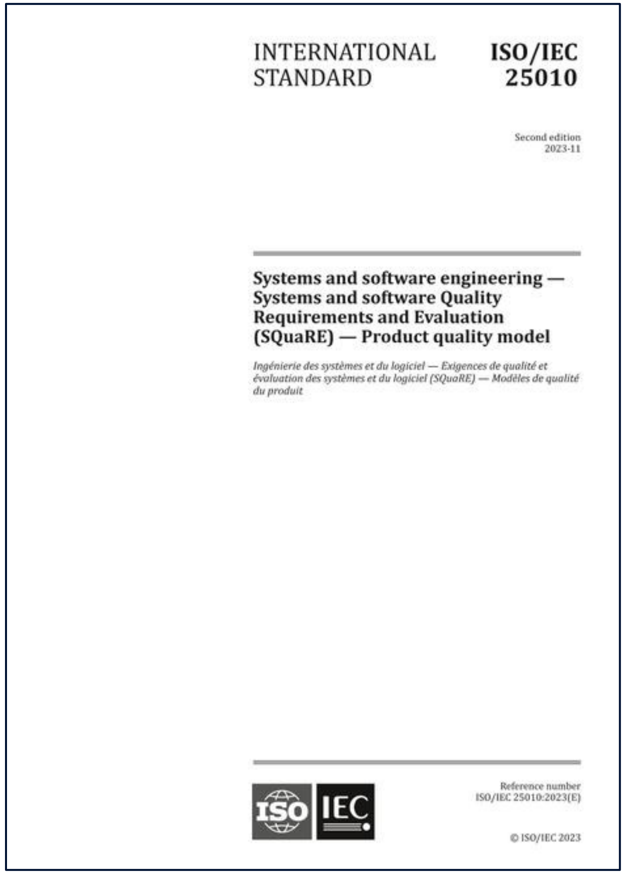
- **penetration tests,**
- **data encryption**
- **security requirements (OWASP ASVS)**
- **unsupported software**

Impact is high, companies see vulnerable or unsupported third-party dependencies as uncompliant and leads to pressure to change including in M&A.

Until now, MD had to sign that he acknowledges the risks, in the future fines can be applied for not addressing the issues.

2 Measuring

Measuring software quality



INTERNATIONAL STANDARD

ISO/IEC 25010

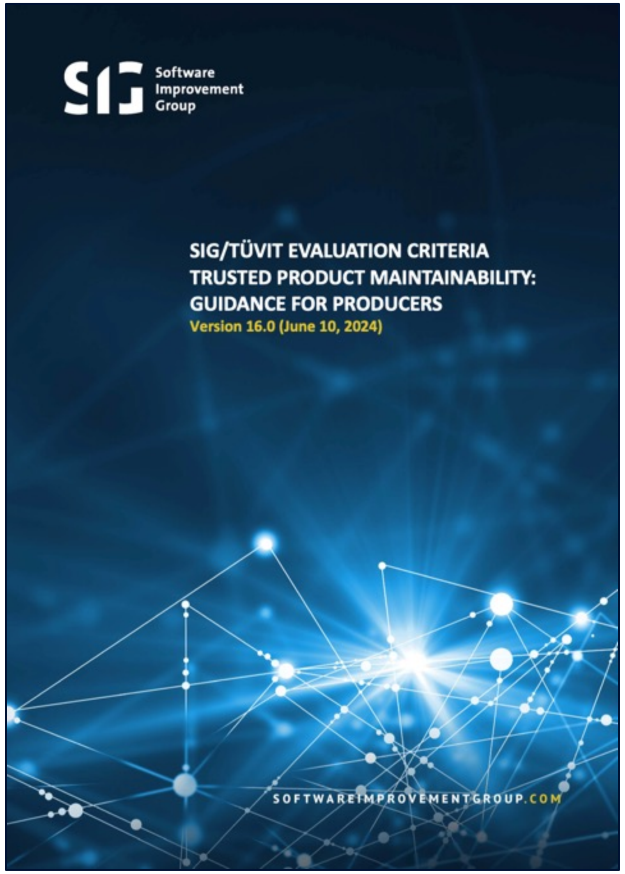
Second edition
2023-11

Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model

Ingénierie des systèmes et du logiciel — Exigences de qualité et évaluation des systèmes et du logiciel (SQuaRE) — Modèles de qualité du produit

Reference number
ISO/IEC 25010:2023(E)

© ISO/IEC 2023

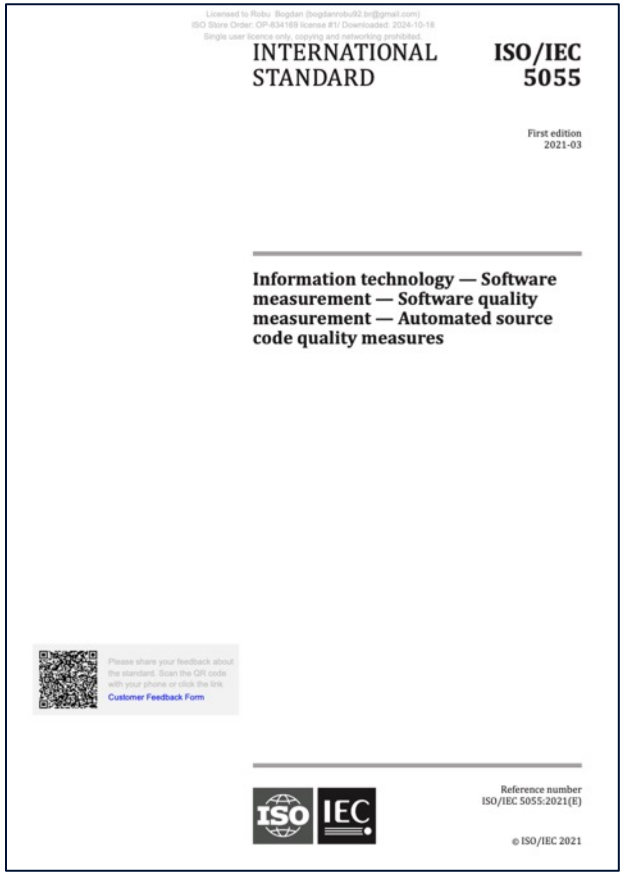


Software Improvement Group

SIG/TÜVIT EVALUATION CRITERIA
TRUSTED PRODUCT MAINTAINABILITY:
GUIDANCE FOR PRODUCERS

Version 16.0 (June 10, 2024)

SOFTWAREIMPROVEMENTGROUP.COM



Licensed to Robu Bogdan (bogdanrobu92.br@gmail.com)
ISO Store Order: OP-834169 license #1/ Downloaded: 2024-10-18
Single user licence only, copying and networking prohibited.

INTERNATIONAL STANDARD

ISO/IEC 5055

First edition
2021-03

Information technology — Software measurement — Software quality measurement — Automated source code quality measures

Please share your feedback about the standard. Scan the QR code with your phone or click the link
Customer Feedback Form

Reference number
ISO/IEC 5055:2021(E)

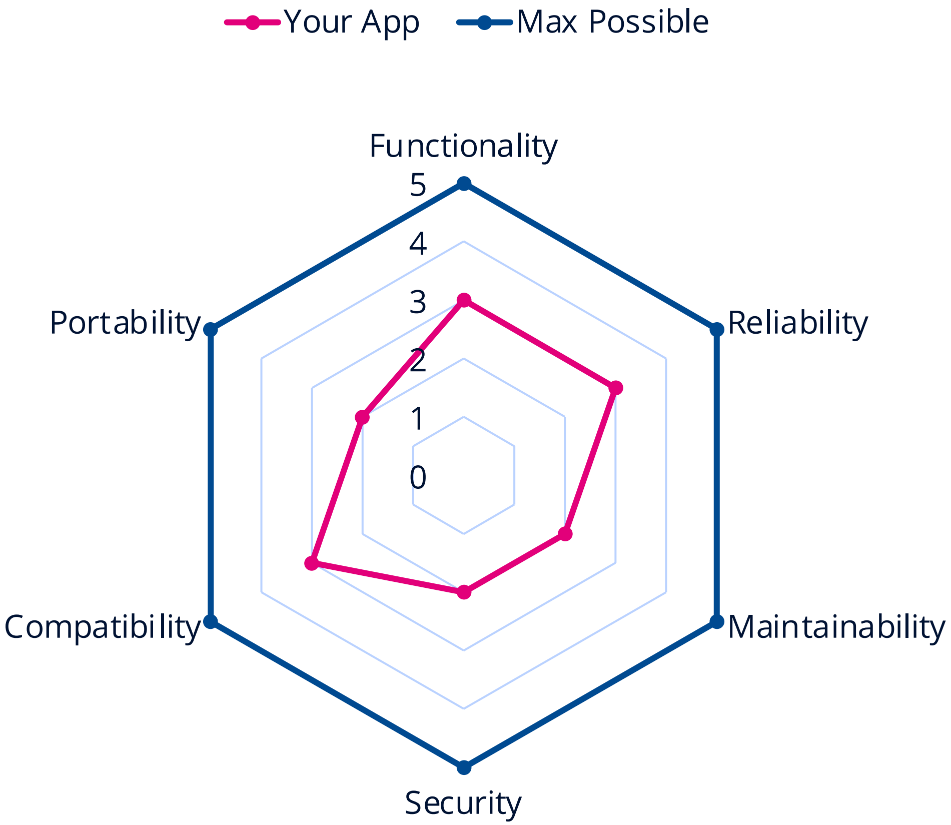
© ISO/IEC 2021

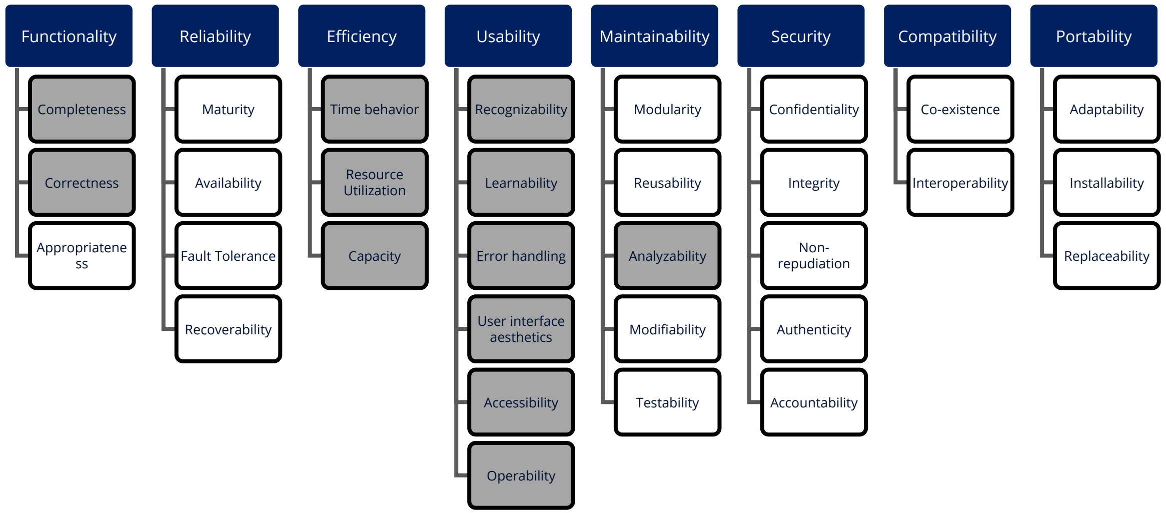
https://www.softwareimprovementgroup.com/wp-content/uploads/SIG-TUViT-Evaluation-Criteria-Trusted-Product-Maintainability-Guidance-for-producers.pdf

https://www.iso.org/standard/80623.html https://www.iso.org/standard/78176.html

ISO / IEC 25010:2011 Product Quality Model

Overall Rating: ★★☆☆☆

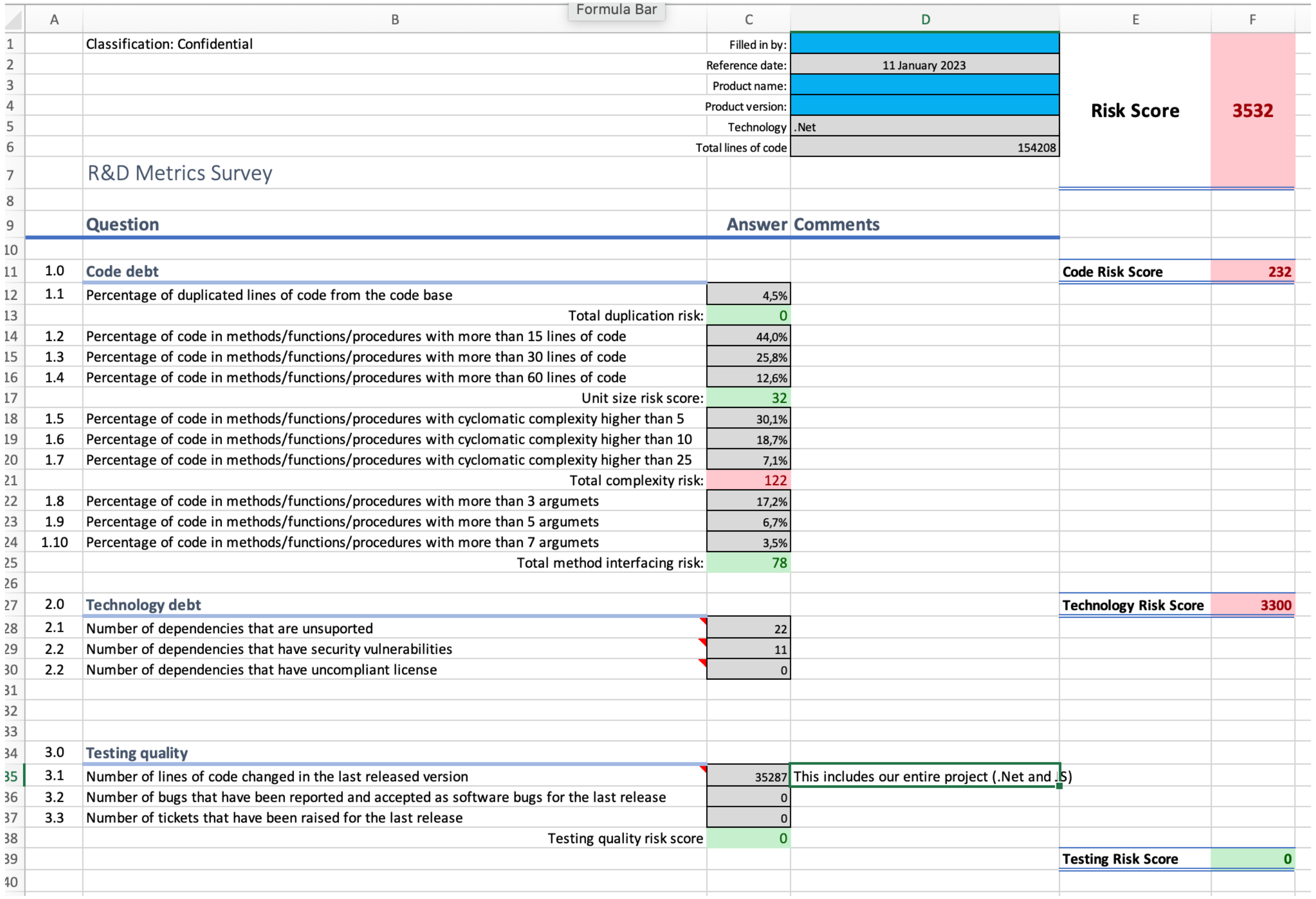




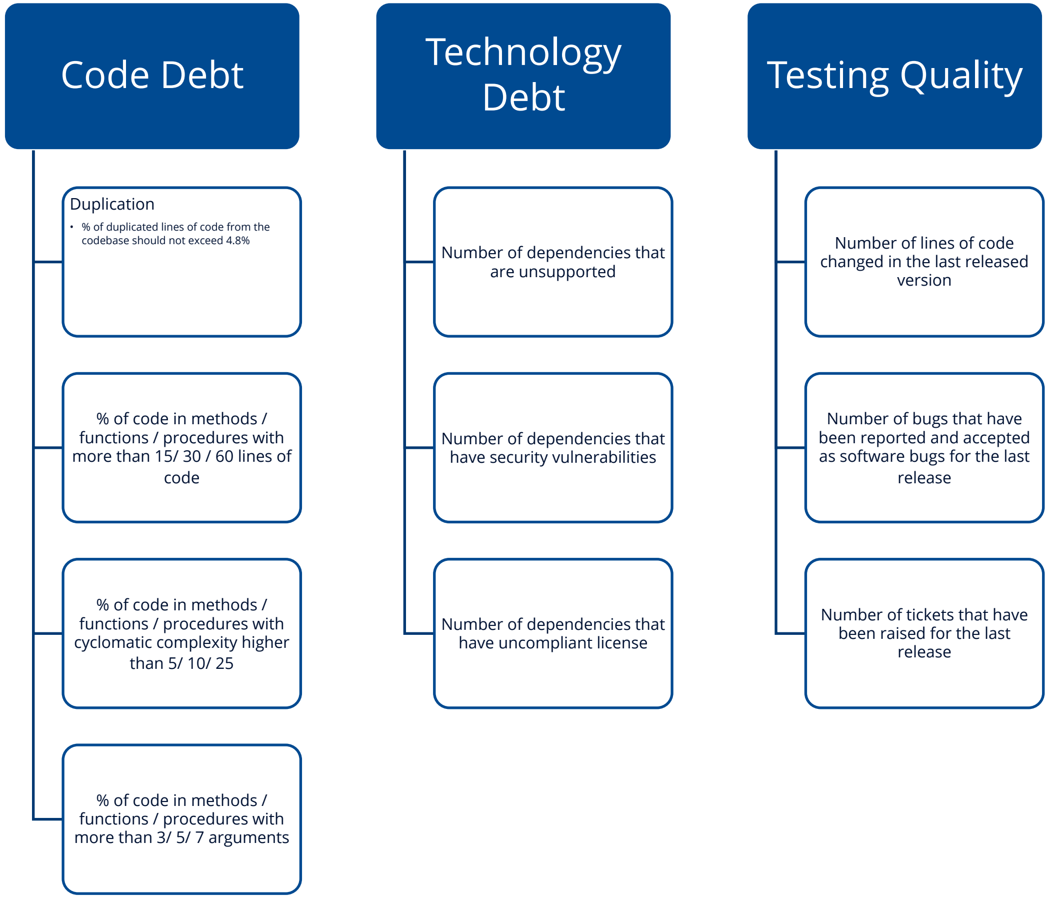
- Functionality
  - Completeness
  - Correctness
  - Appropriateness
- Reliability
  - Maturity
  - Availability
  - Fault Tolerance
  - Recoverability
- Efficiency
  - Time behavior
  - Resource Utilization
  - Capacity
- Usability
  - Recognizability
  - Learnability
  - Error handling
  - User interface aesthetics
  - Accessibility
  - Operability
- Maintainability
  - Modularity
  - Reusability
  - Analyzability
  - Modifiability
  - Testability
- Security
  - Confidentiality
  - Integrity
  - Non-repudiation
  - Authenticity
  - Accountability
- Compatibility
  - Co-existence
  - Interoperability
- Portability
  - Adaptability
  - Installability
  - Replaceability

yonder

R&D Metrics



	A	B	C	D	E	F
1		Classification: Confidential	Filled in by:			
2			Reference date:	11 January 2023		
3			Product name:			
4			Product version:		Risk Score	3532
5			Technology	.Net		
6			Total lines of code	154208		
7		R&D Metrics Survey				
8						
9		Question	Answer	Comments		
10						
11	1.0	Code debt			Code Risk Score	232
12	1.1	Percentage of duplicated lines of code from the code base	4,5%			
13		Total duplication risk:	0			
14	1.2	Percentage of code in methods/functions/procedures with more than 15 lines of code	44,0%			
15	1.3	Percentage of code in methods/functions/procedures with more than 30 lines of code	25,8%			
16	1.4	Percentage of code in methods/functions/procedures with more than 60 lines of code	12,6%			
17		Unit size risk score:	32			
18	1.5	Percentage of code in methods/functions/procedures with cyclomatic complexity higher than 5	30,1%			
19	1.6	Percentage of code in methods/functions/procedures with cyclomatic complexity higher than 10	18,7%			
20	1.7	Percentage of code in methods/functions/procedures with cyclomatic complexity higher than 25	7,1%			
21		Total complexity risk:	122			
22	1.8	Percentage of code in methods/functions/procedures with more than 3 argumets	17,2%			
23	1.9	Percentage of code in methods/functions/procedures with more than 5 argumets	6,7%			
24	1.10	Percentage of code in methods/functions/procedures with more than 7 argumets	3,5%			
25		Total method interfacing risk:	78			
26						
27	2.0	Technology debt			Technology Risk Score	3300
28	2.1	Number of dependencies that are unsuported	22			
29	2.2	Number of dependencies that have security vulnerabilities	11			
30	2.2	Number of dependencies that have uncompliant license	0			
31						
32						
33						
34	3.0	Testing quality				
35	3.1	Number of lines of code changed in the last released version	35287	This includes our entire project (.Net and JS)		
36	3.2	Number of bugs that have been reported and accepted as software bugs for the last release	0			
37	3.3	Number of tickets that have been raised for the last release	0			
38		Testing quality risk score	0			
39					Testing Risk Score	0
40						



Code Debt

Duplication

- % of duplicated lines of code from the codebase should not exceed 4.8%

% of code in methods / functions / procedures with more than 15/ 30 / 60 lines of code

% of code in methods / functions / procedures with cyclomatic complexity higher than 5/ 10/ 25

% of code in methods / functions / procedures with more than 3/ 5/ 7 arguments

Technology Debt

Number of dependencies that are unsupported

Number of dependencies that have security vulnerabilities

Number of dependencies that have uncompliant license

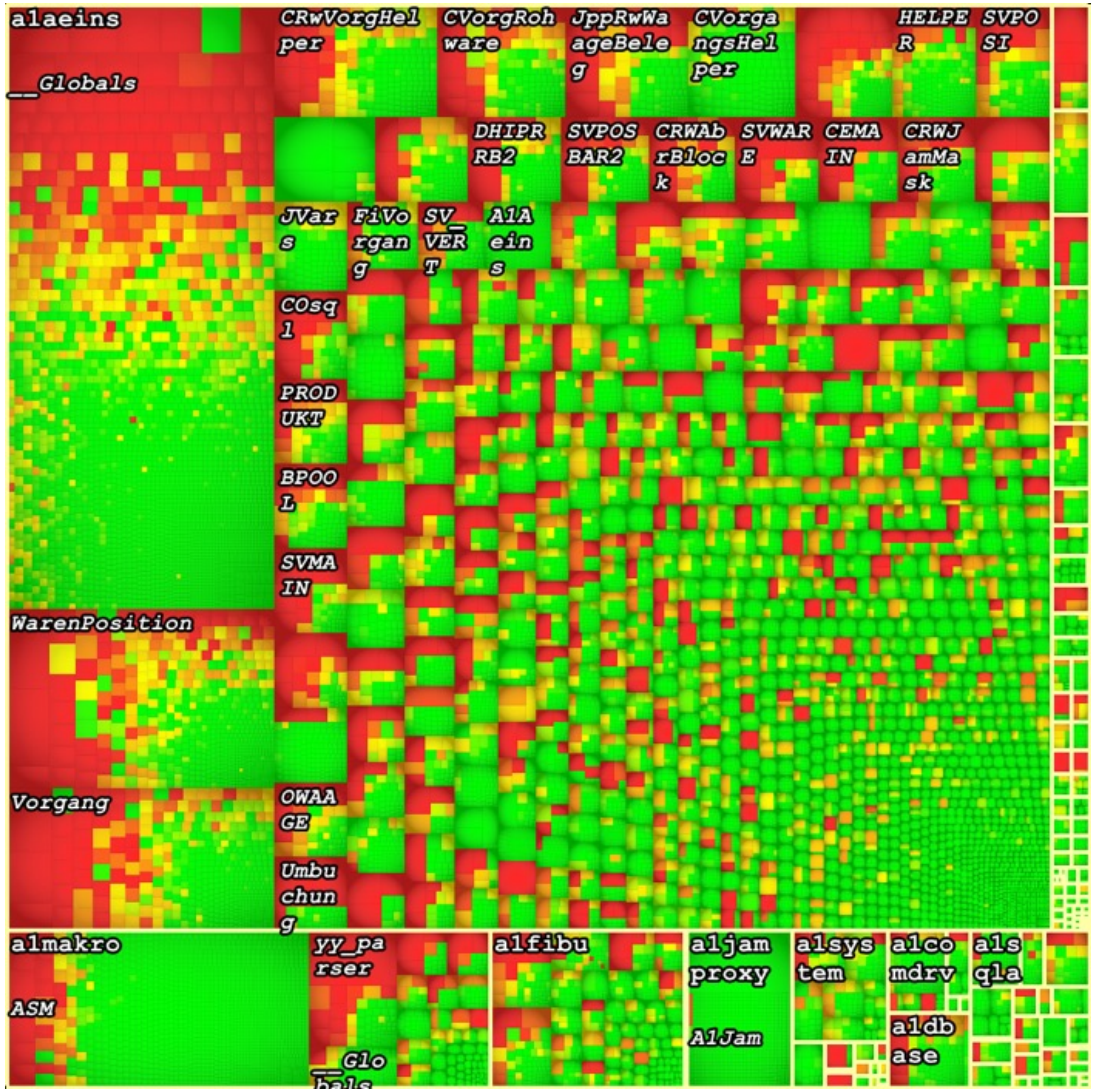
Testing Quality

Number of lines of code changed in the last released version

Number of bugs that have been reported and accepted as software bugs for the last release

Number of tickets that have been raised for the last release

How code debt looks like?

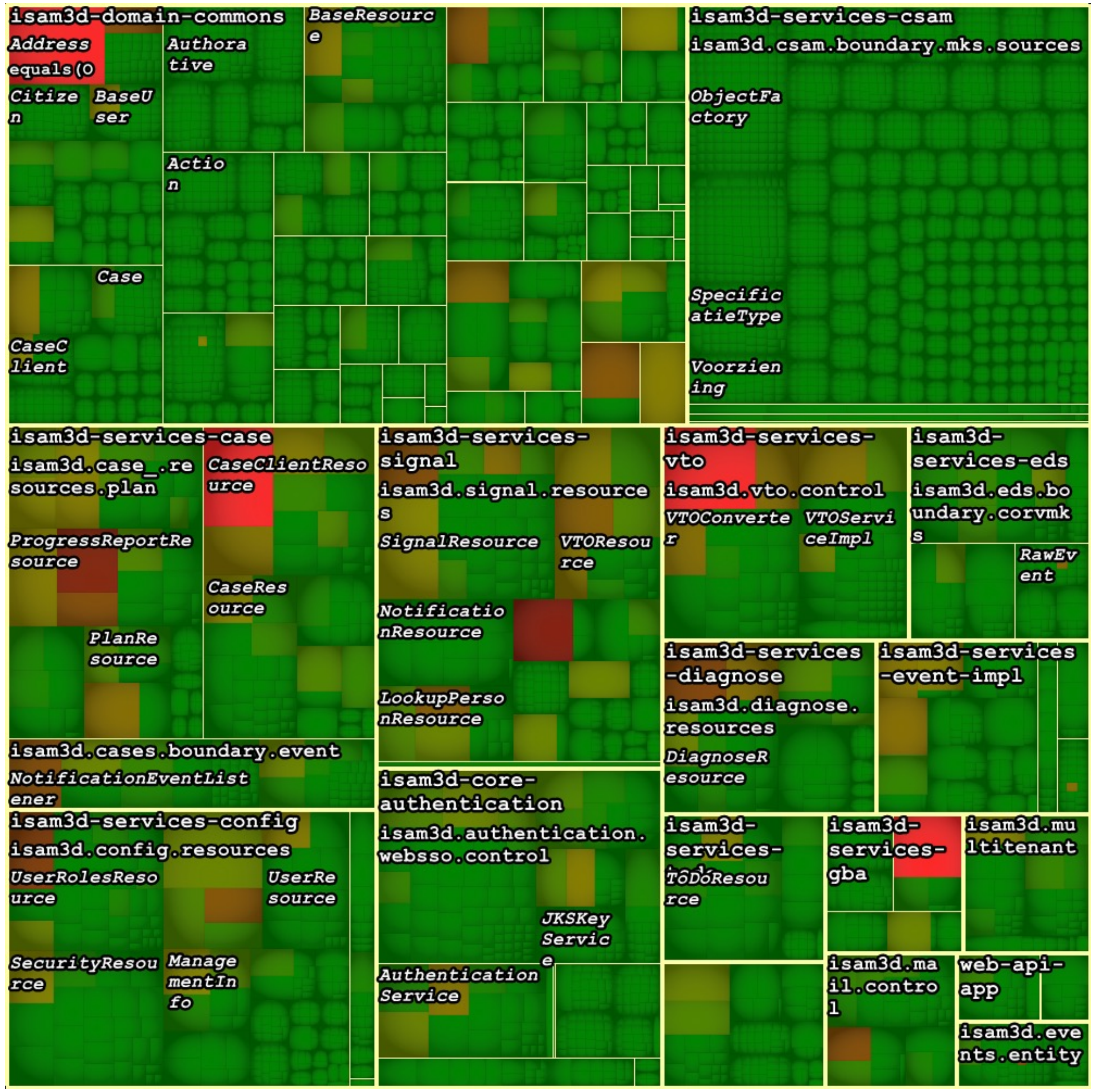


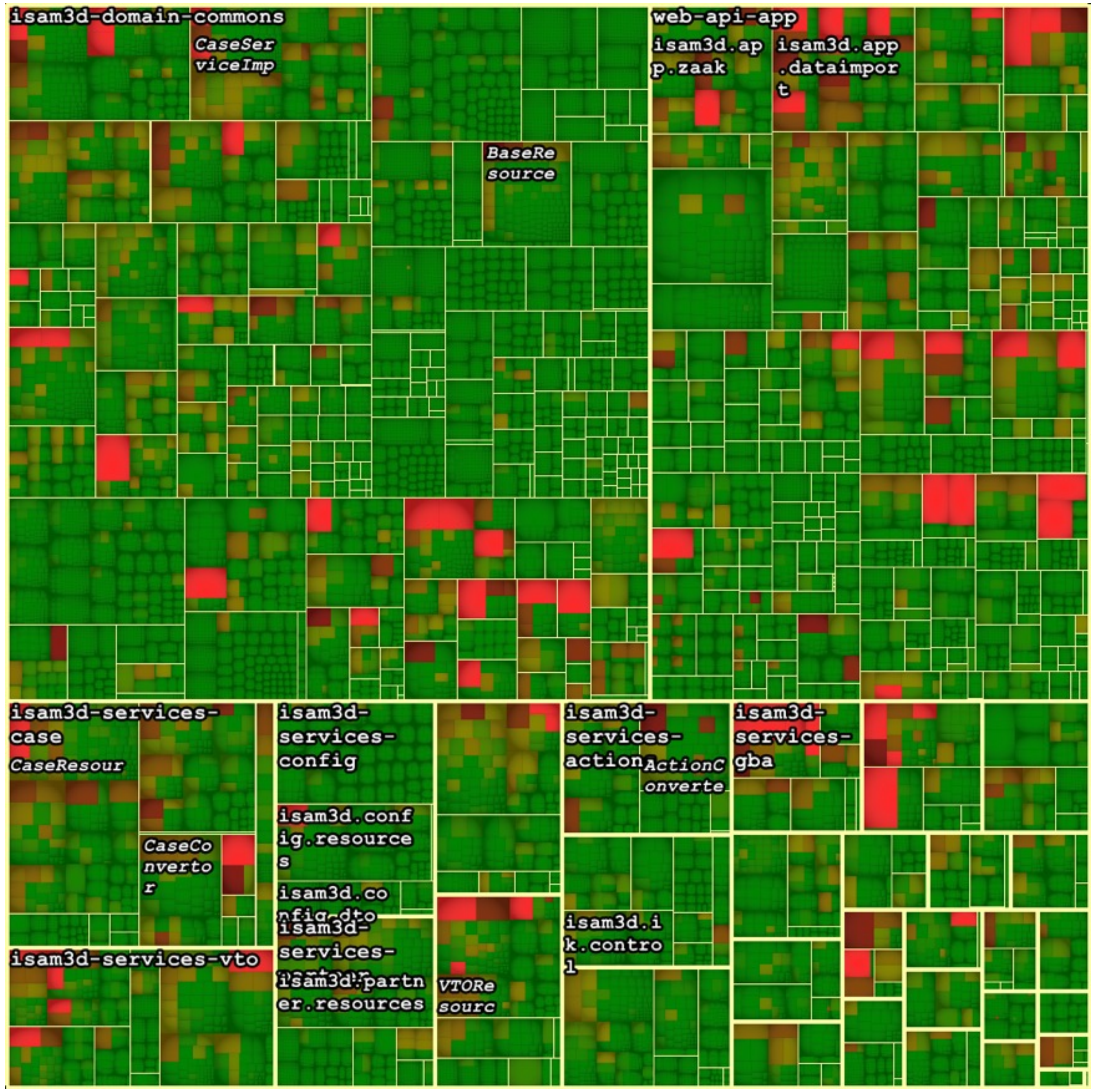
Red: more than 25 decisions / method

Yellow: more than 10 decision / method

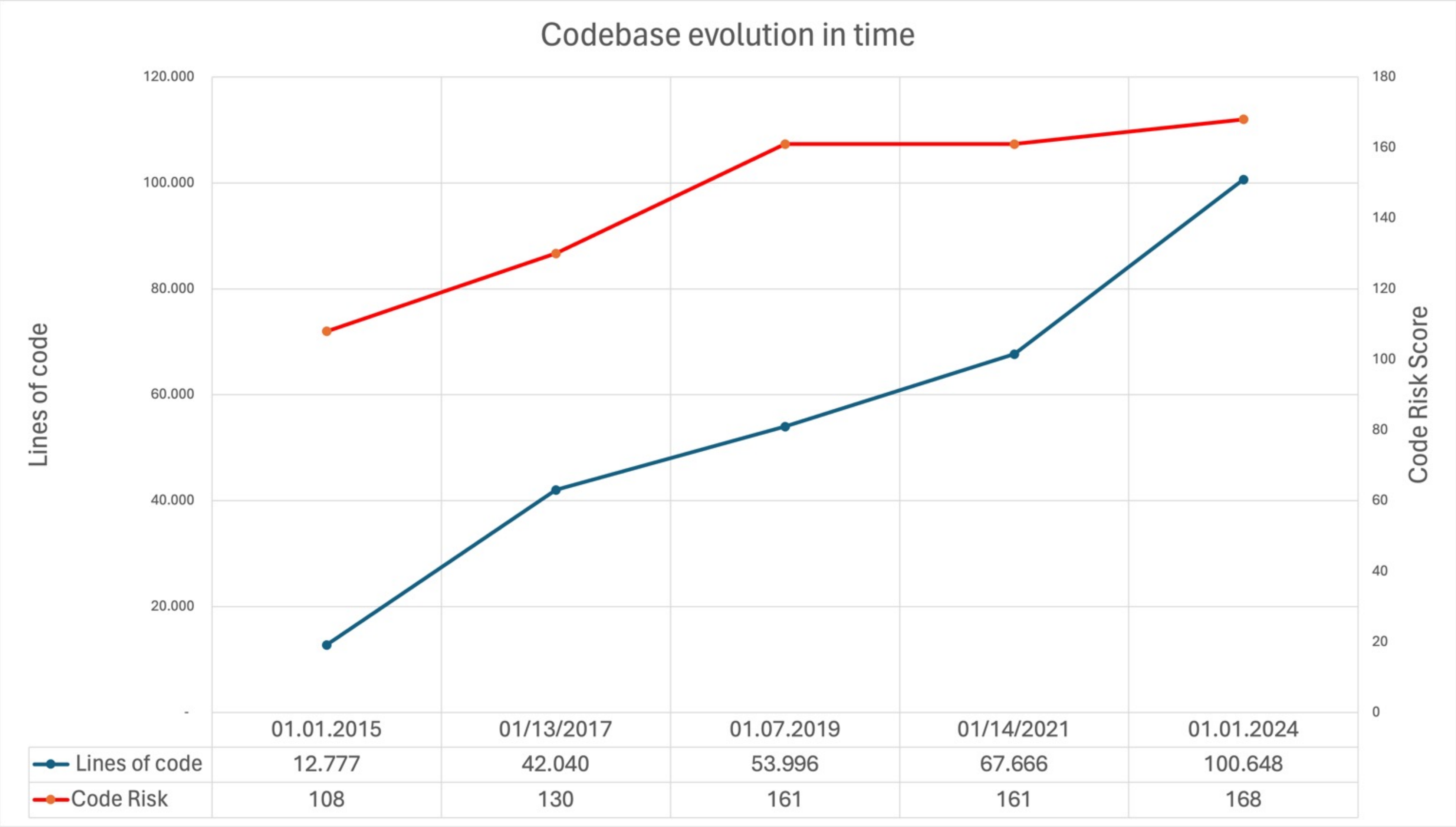
Green: more than 5 decision / method

Looking at a long running project





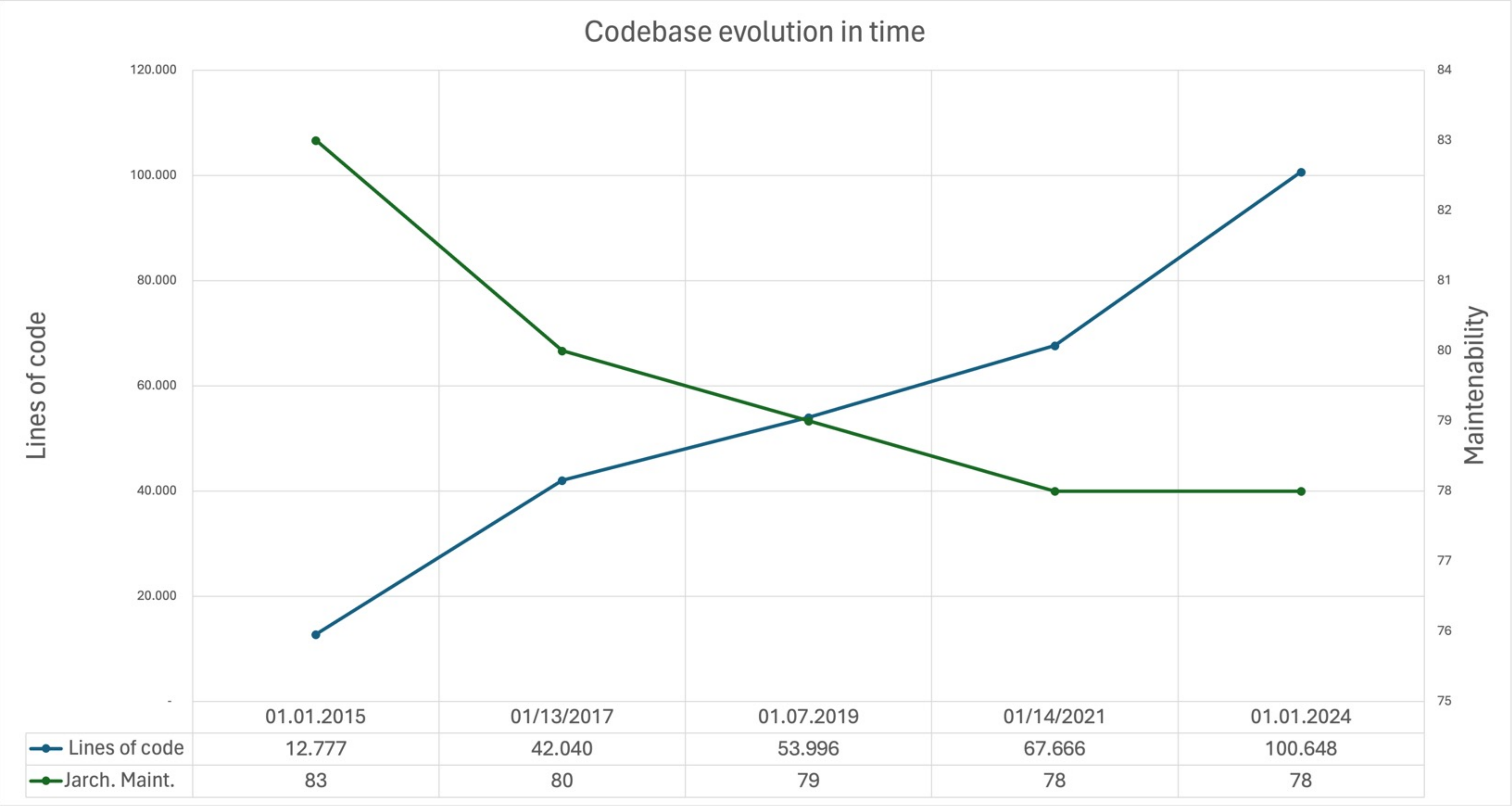
Looking at a long running project



Codebase evolution in time

	01.01.2015	01/13/2017	01.07.2019	01/14/2021	01.01.2024
Lines of code	12.777	42.040	53.996	67.666	100.648
Code Risk	108	130	161	161	168

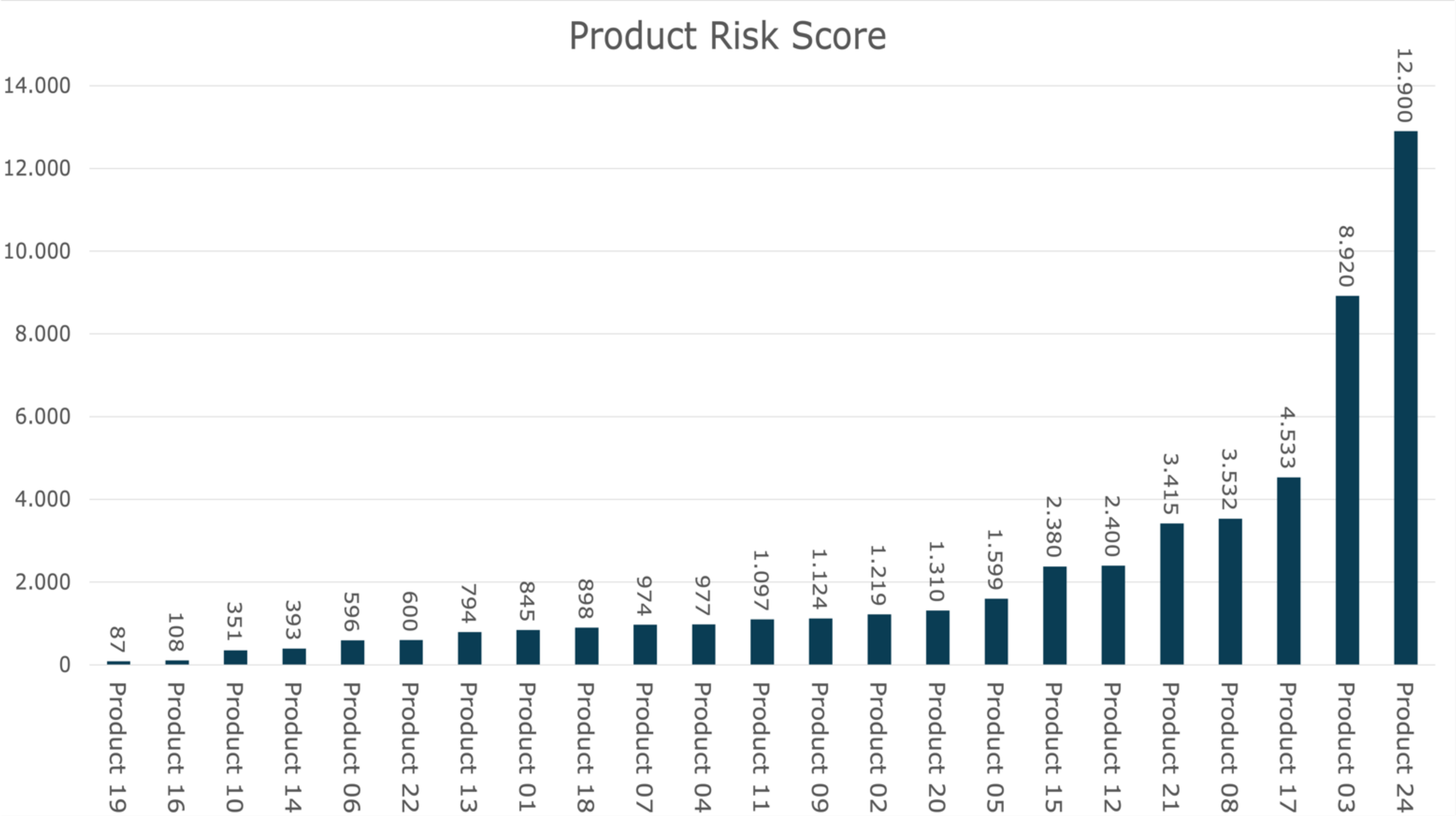
Code debt – why is it important



	01.01.2015	01/13/2017	01.07.2019	01/14/2021	01.01.2024
Lines of code	12.777	42.040	53.996	67.666	100.648
Jarch. Maint.	83	80	79	78	78

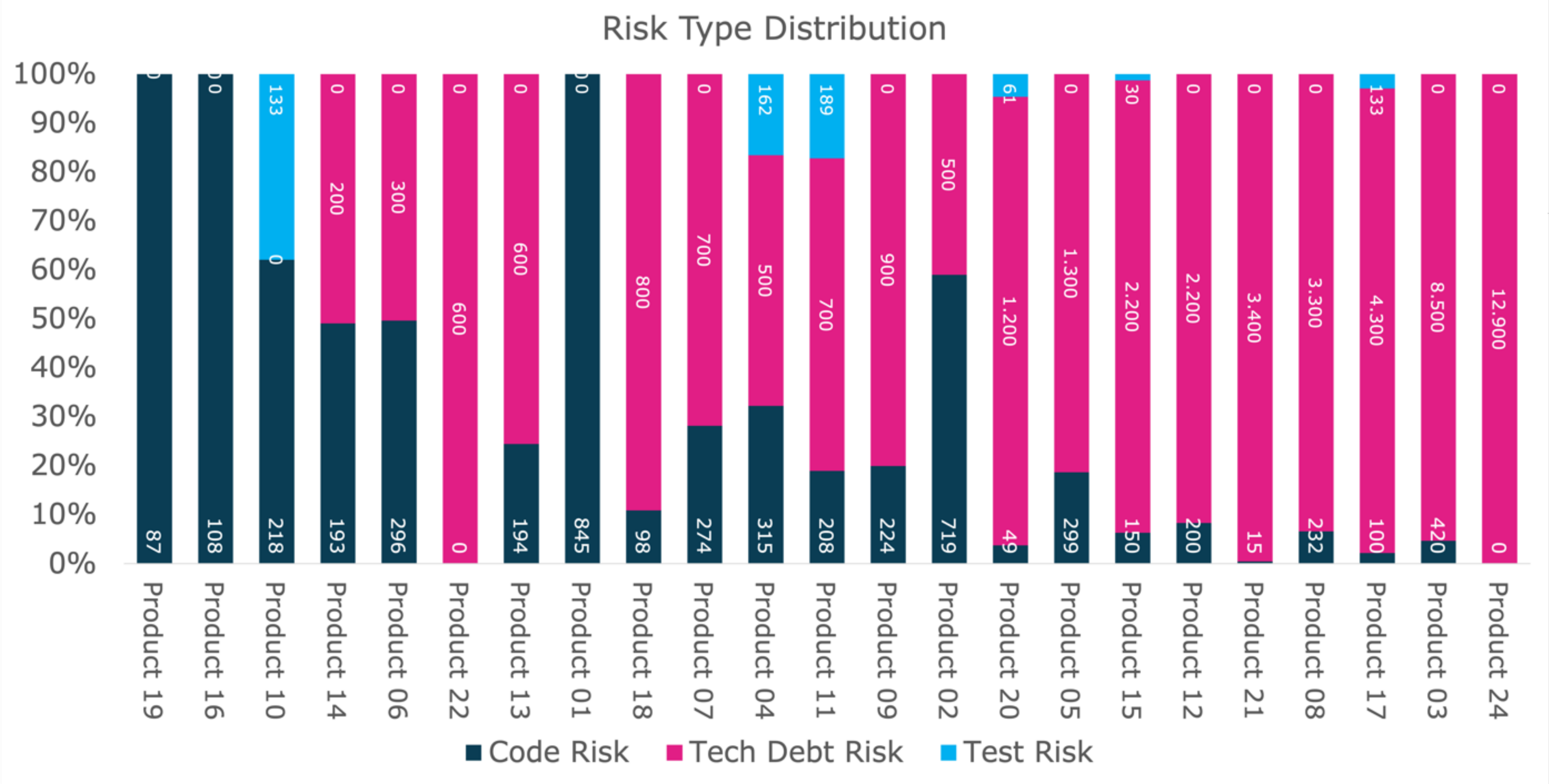
R&D Metrics

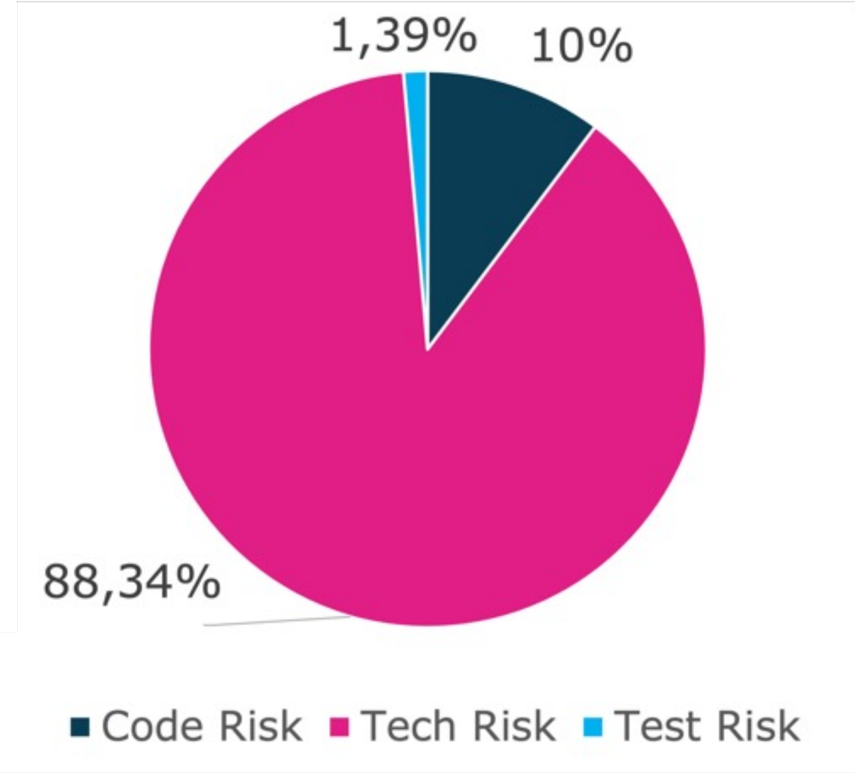
24 products measured
2 portfolios
None had zero risk



R&D Metrics

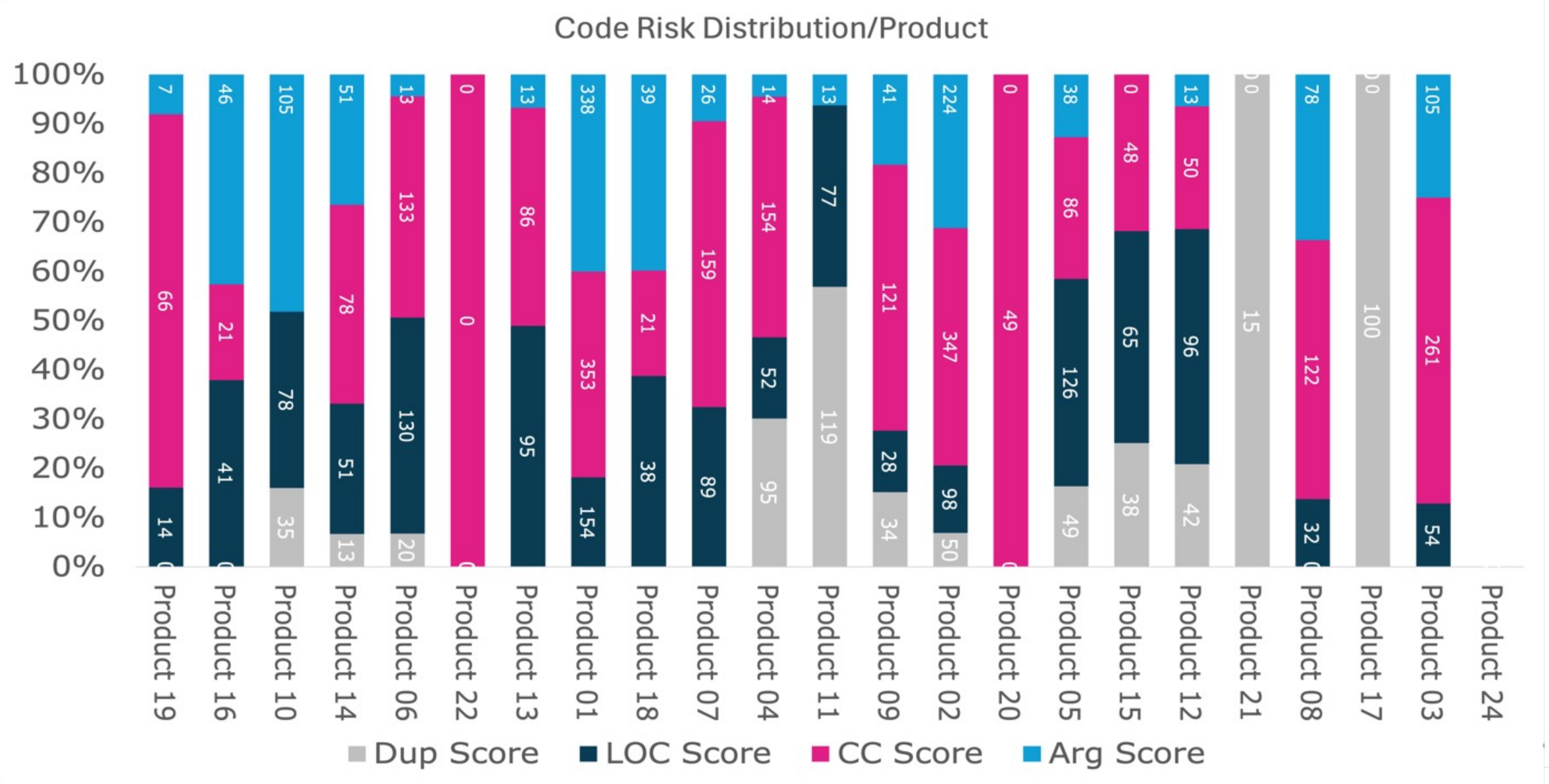
Tech Risk – unsupported dependencies or dependencies with security vulnerabilities, the biggest risk.

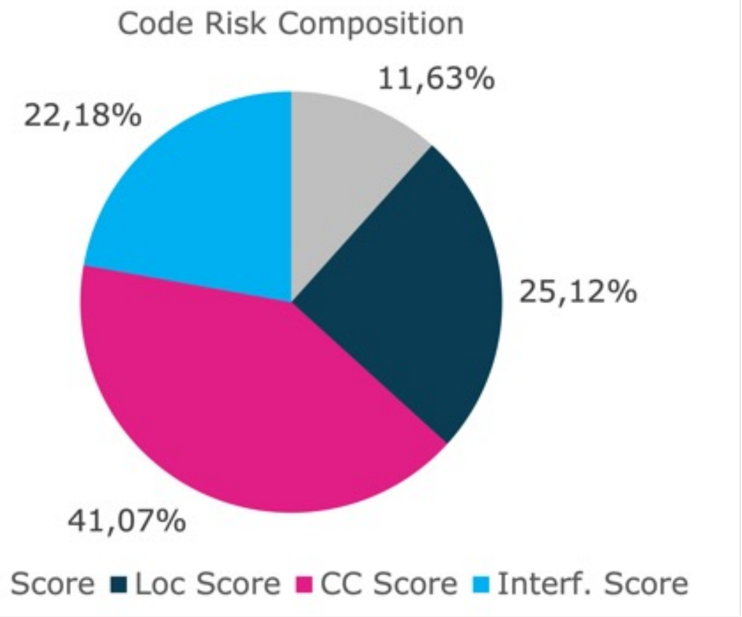

Risk Type Distribution



R&D Metrics

Cyclomatic complexity, high number of decisions, the biggest problem.





yonder

Does these results match your expectations?

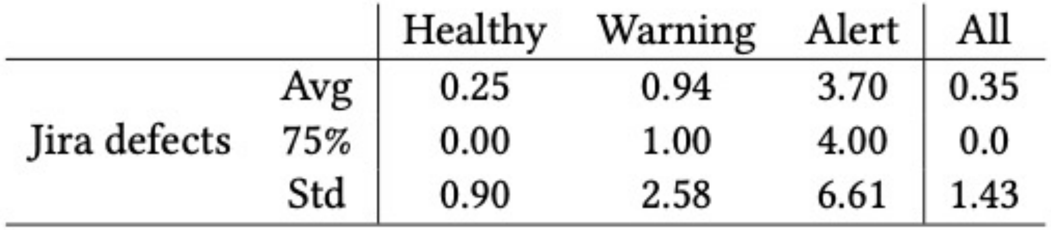
2 Impact

What is the impact of code debt is in your portfolio?

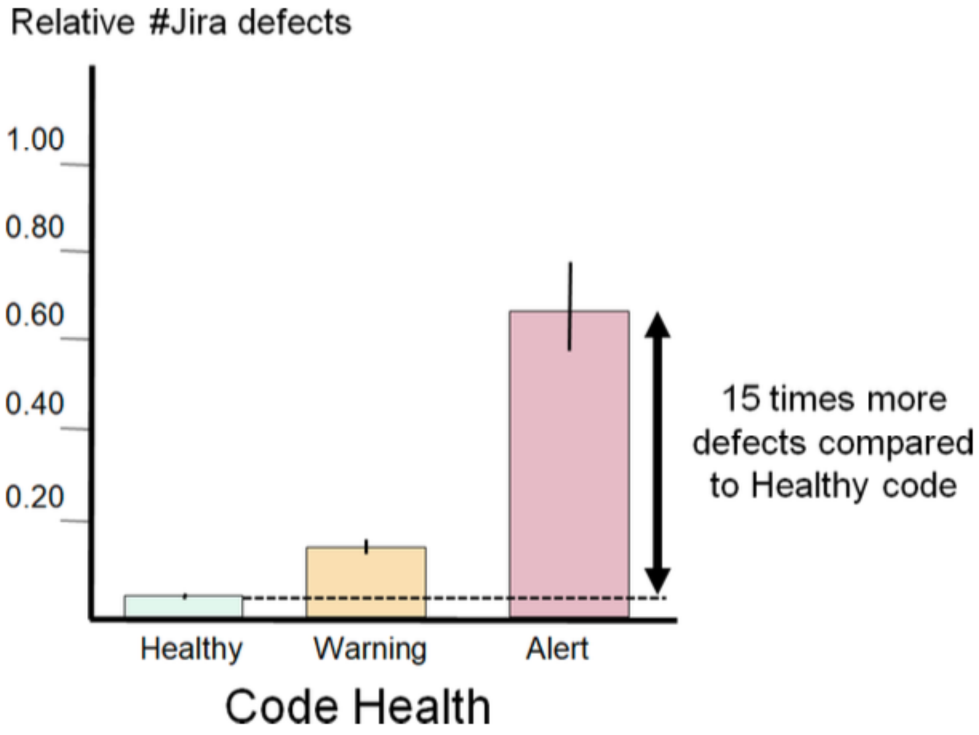
yonder

The impact of code debt

Table 2: Average number of Jira defects per file for each Code Health category.

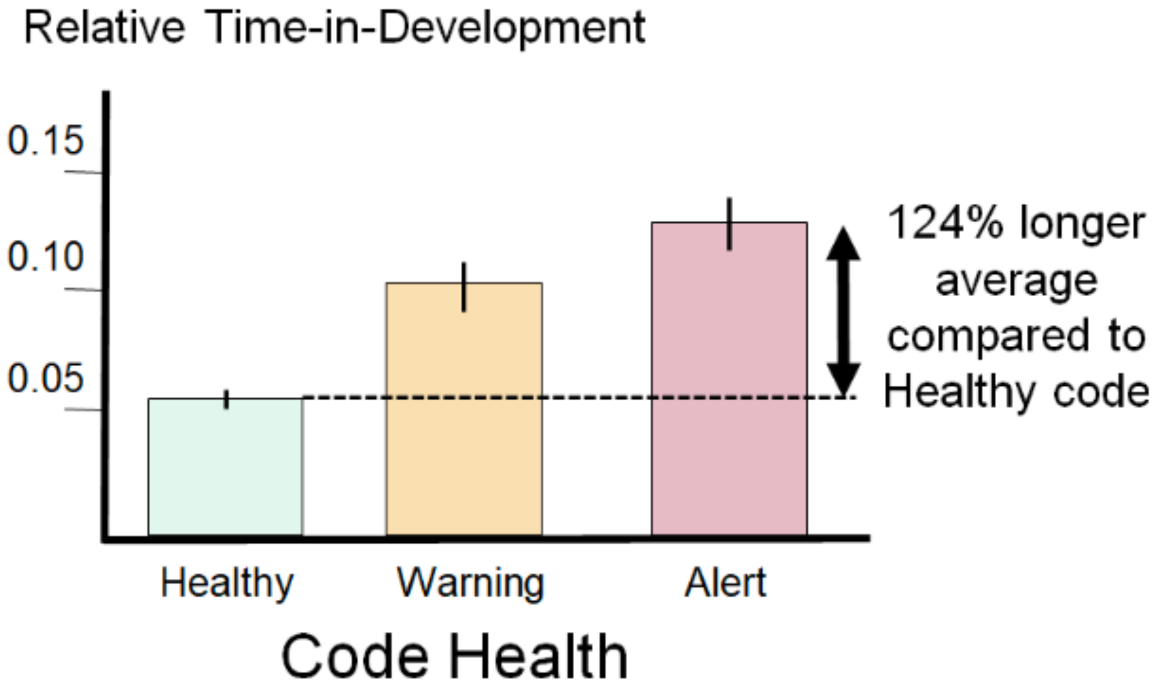


		Healthy	Warning	Alert	All
Jira defects	Avg	0.25	0.94	3.70	0.35
	75%	0.00	1.00	4.00	0.0
	Std	0.90	2.58	6.61	1.43



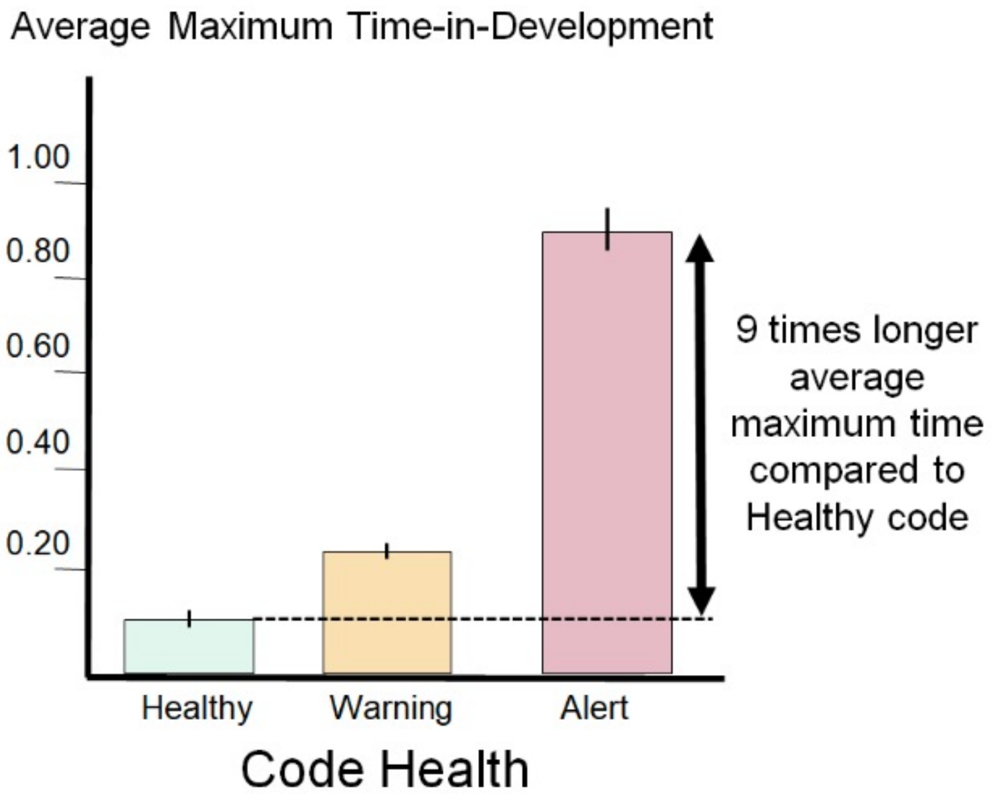
https://codescene.com/hubfs/web_docs/Business-impact-of-code-quality.pdf
https://arxiv.org/abs/2203.04374

The impact of code debt



124% longer average compared to Healthy code

Figure 8: Average Time-in-Development (scaled) for resolving a Jira issue per file. The standard errors are depicted as vertical lines.



9 times longer average maximum time compared to Healthy code

Figure 9: Average maximum Time-in-Development (scaled) for resolving a Jira issue per file. The standard errors are shown as vertical lines.

https://codescene.com/hubfs/web_docs/Business-impact-of-code-quality.pdf
https://arxiv.org/abs/2203.04374

Tracking planned vs. unplanned work?

yonder

The impact of code debt

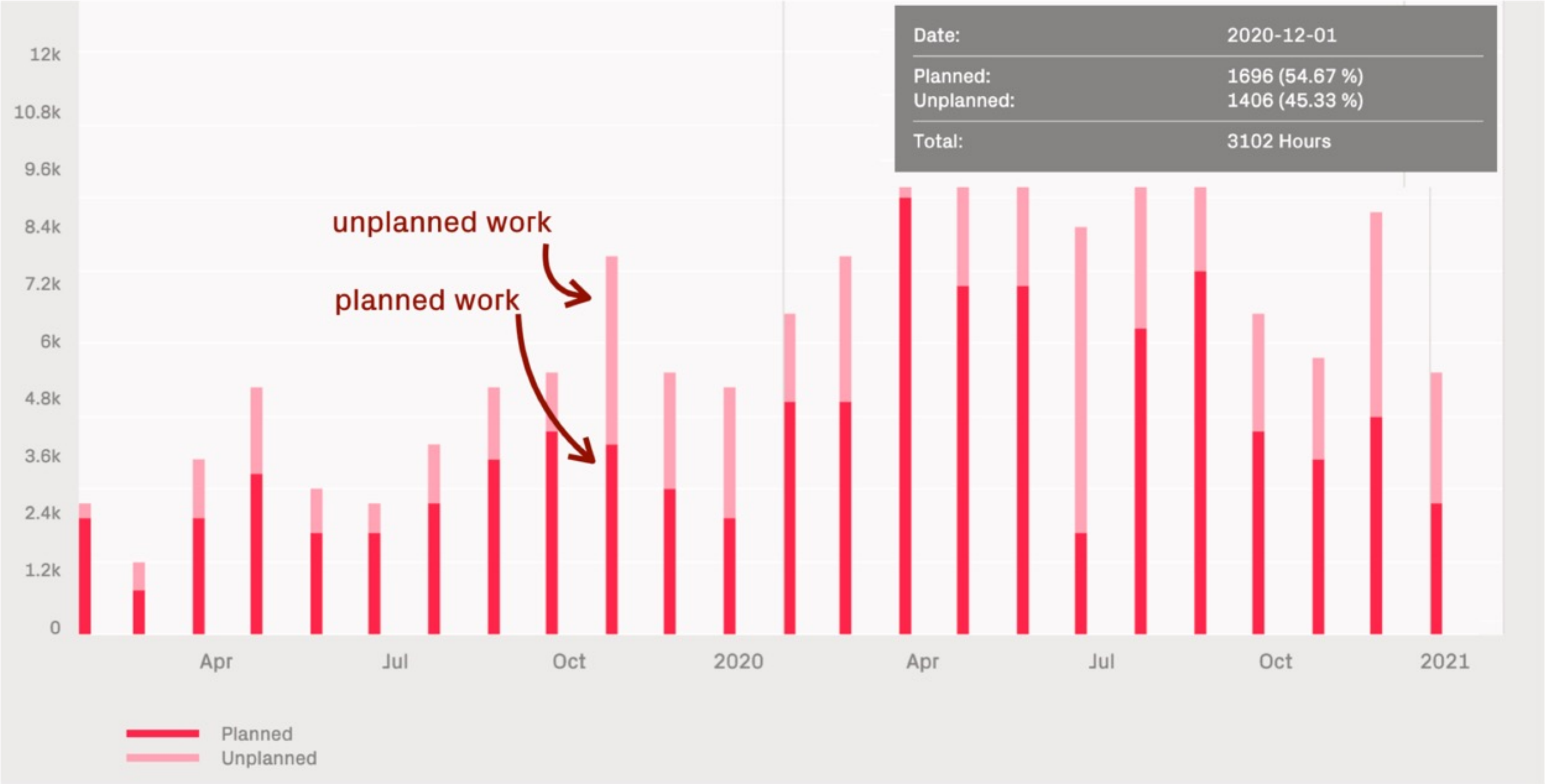
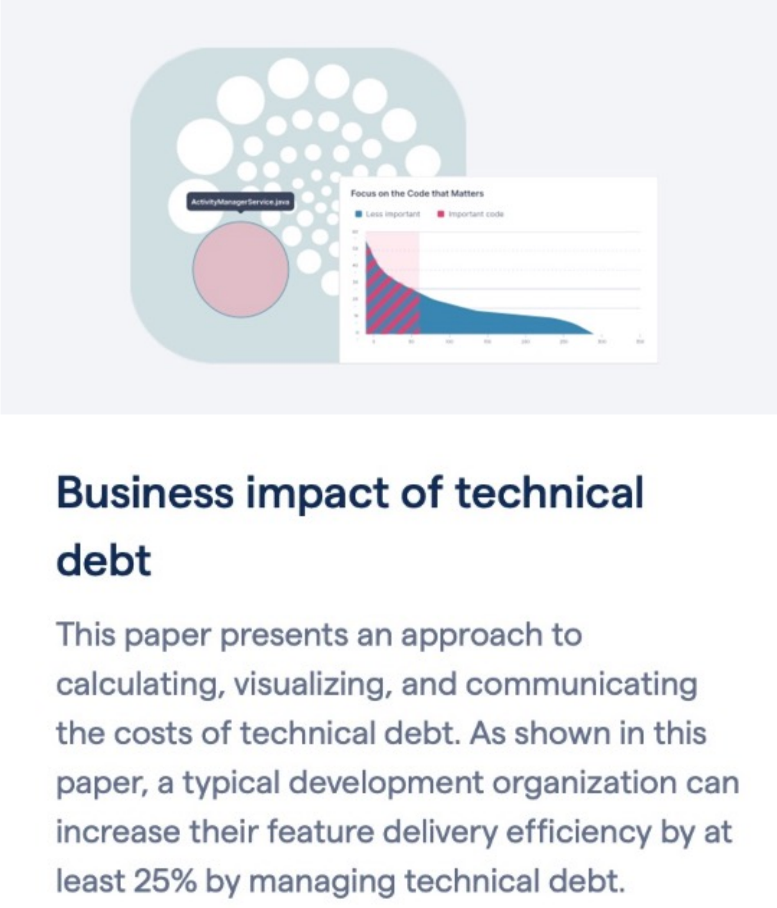


Figure 2. Trend showing the percentage of Unplanned Work over the past year.
On average, 40-50% of the development time is wasted on unplanned work.

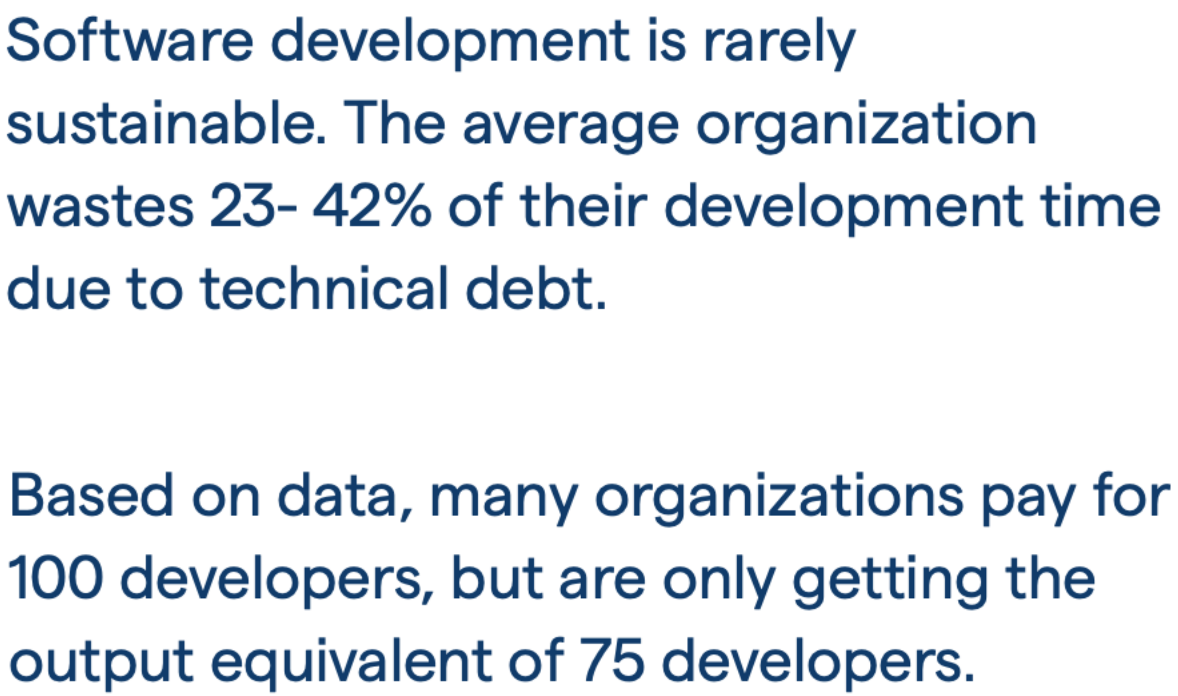
https://codescene.com/hubfs/calculate-business-costs-of-technical-debt.pdf
https://arxiv.org/pdf/2401.13407v1

The impact of code debt



Business impact of technical debt

This paper presents an approach to calculating, visualizing, and communicating the costs of technical debt. As shown in this paper, a typical development organization can increase their feature delivery efficiency by at least 25% by managing technical debt.



Software development is rarely sustainable. The average organization wastes 23- 42% of their development time due to technical debt.

Based on data, many organizations pay for 100 developers, but are only getting the output equivalent of 75 developers.

https://codescene.com/hubfs/calculate-business-costs-of-technical-debt.pdf
https://arxiv.org/pdf/2401.13407v1

The impact of code debt

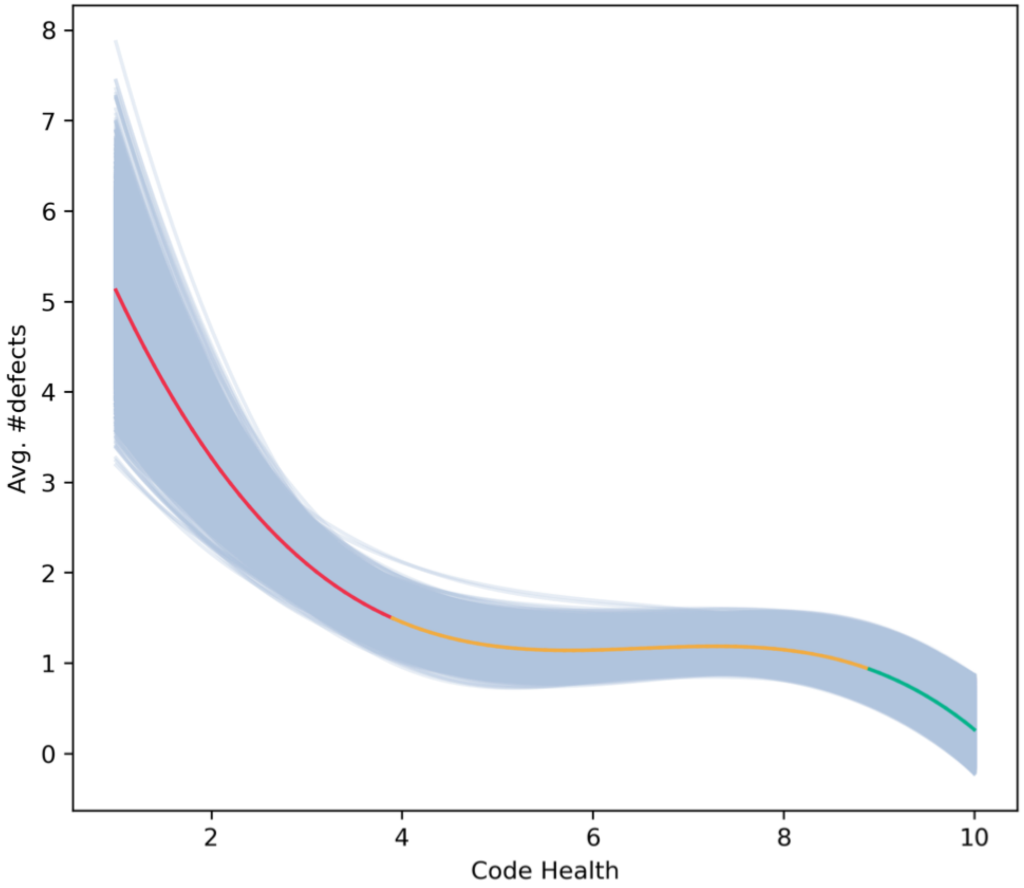


Figure 2: Average defect count per file for different CH.

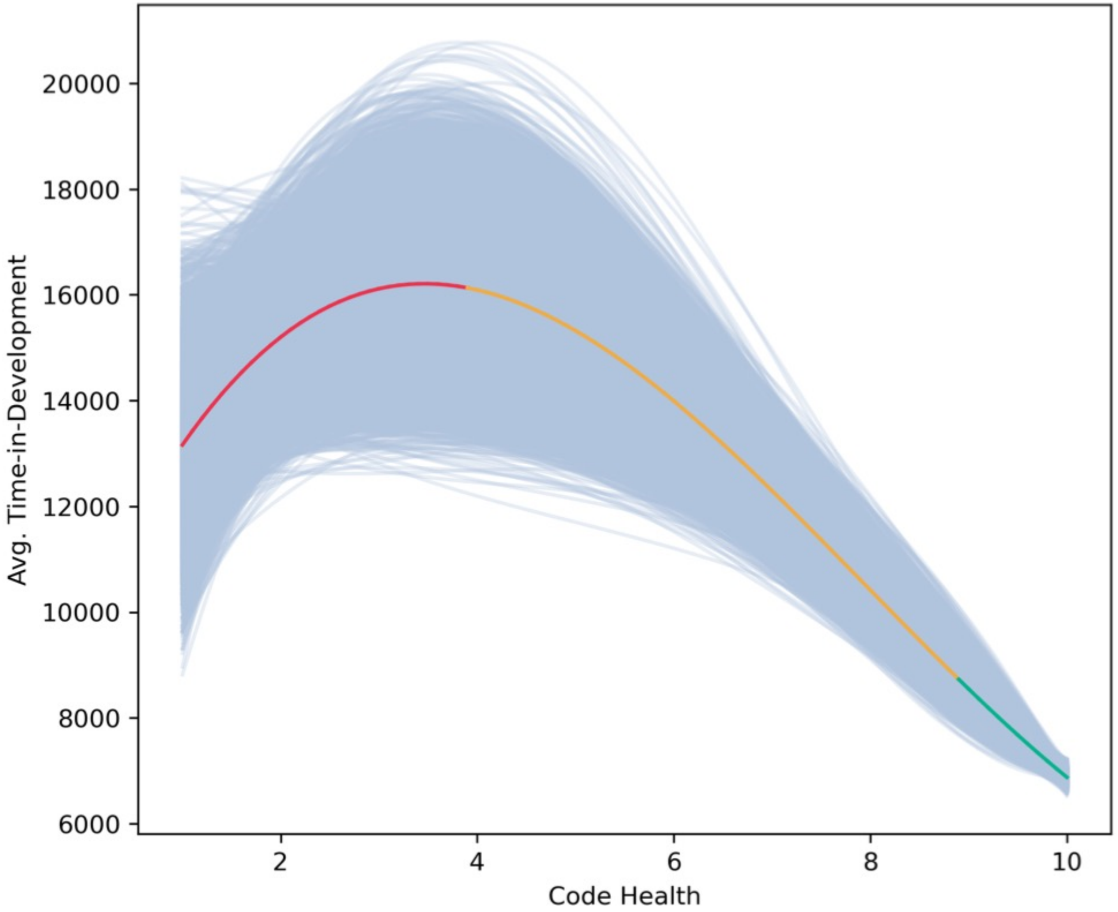
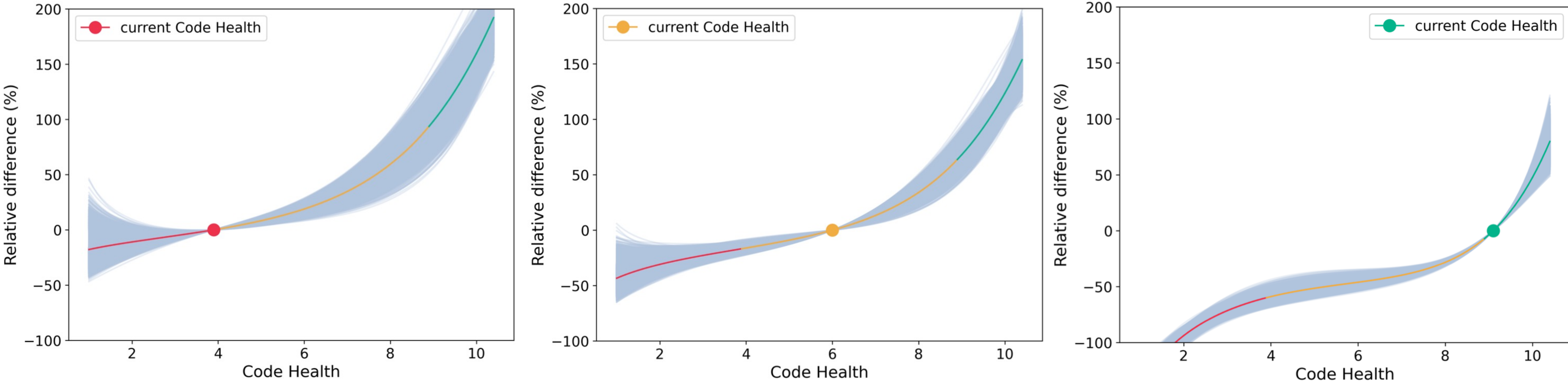


Figure 3: Average Time-in-Dev for resolving issues.

https://codescene.com/hubfs/calculate-business-costs-of-technical-debt.pdf
https://arxiv.org/pdf/2401.13407v1

Increasing, not diminishing returns



(a) Starting point: $CH_0 = 3.9, u = 0.12$

(c) Starting point: $CH_0 = 6.0, u = 0.12$

(f) Starting point: $CH_0 = 9.1, u = 0.25$

https://codescene.com/hubfs/calculate-business-costs-of-technical-debt.pdf
https://arxiv.org/pdf/2401.13407v1

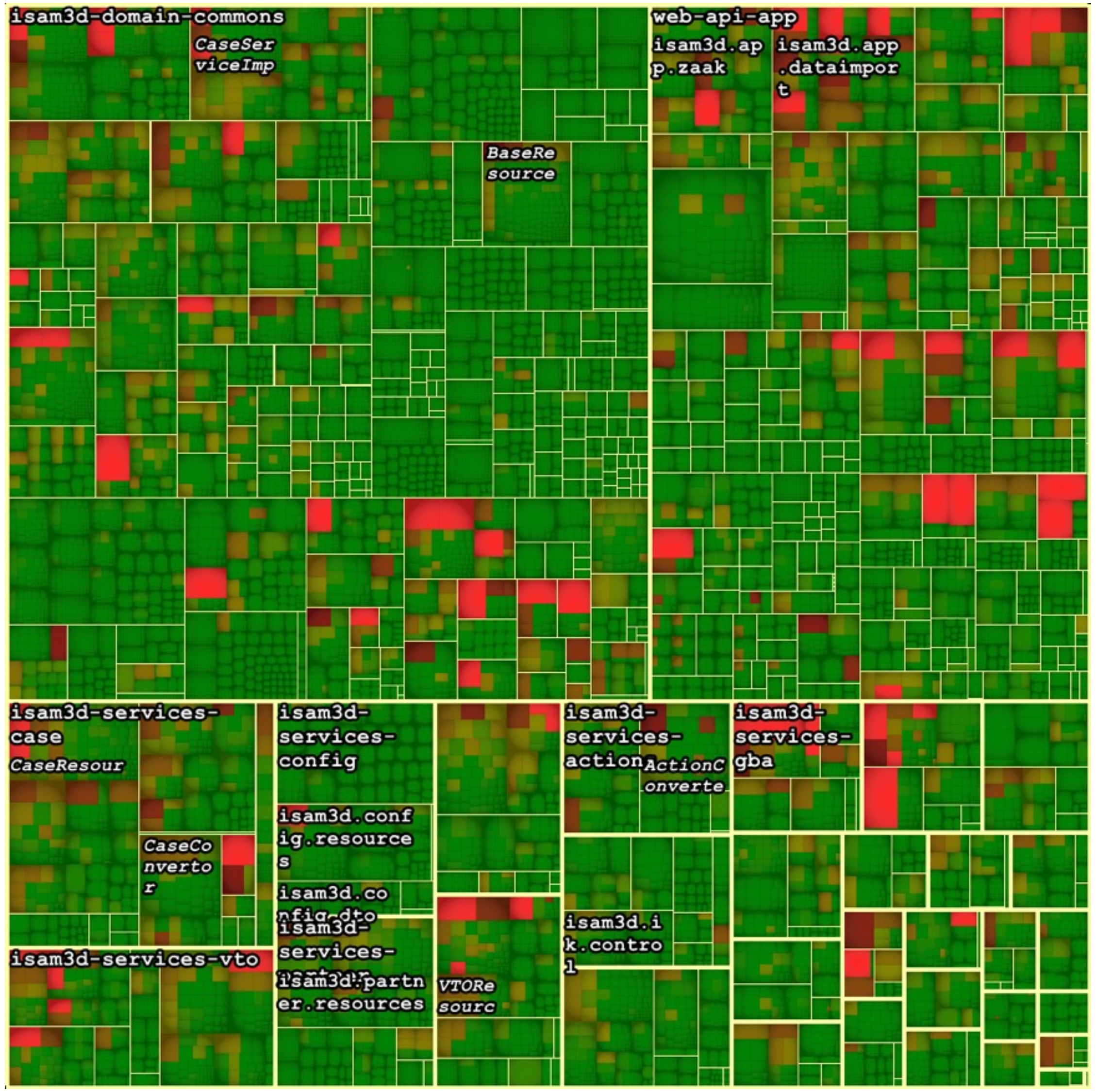
Nice, but what does it mean for a CSI project?

Looking at a long running project

- > 10 years of development
- SAAS multi-tenant
- Team varied between 5-12 FTE over the time
- > 1mil. lines of code (backend)
- Java based

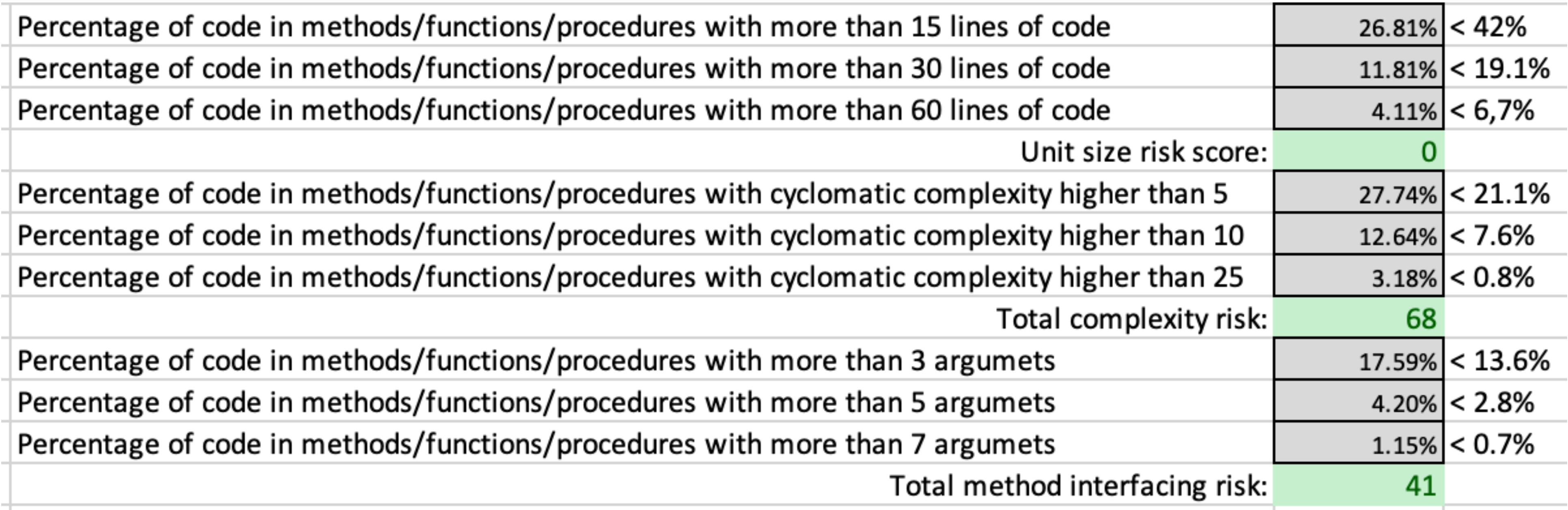
- Jira tickets linked to GIT commits from day 1
- Jira tickets, have effort logged
- (Some) Jira tickets have full time (estimated/actual)

- 2 636 Java Classes
- 29 498 Java Methods
- 3939 Done Bugs



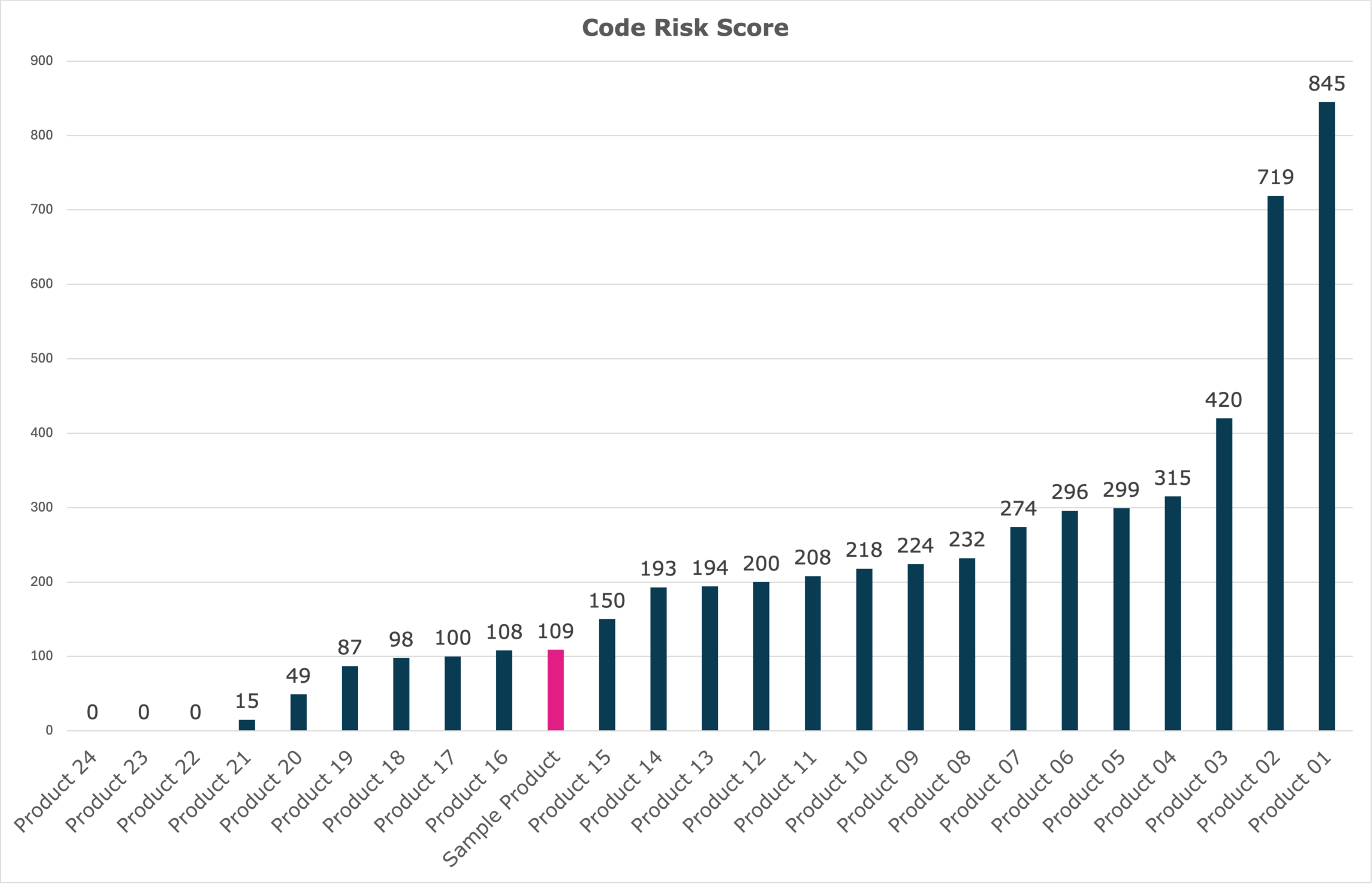
yonder

Code risk not zero

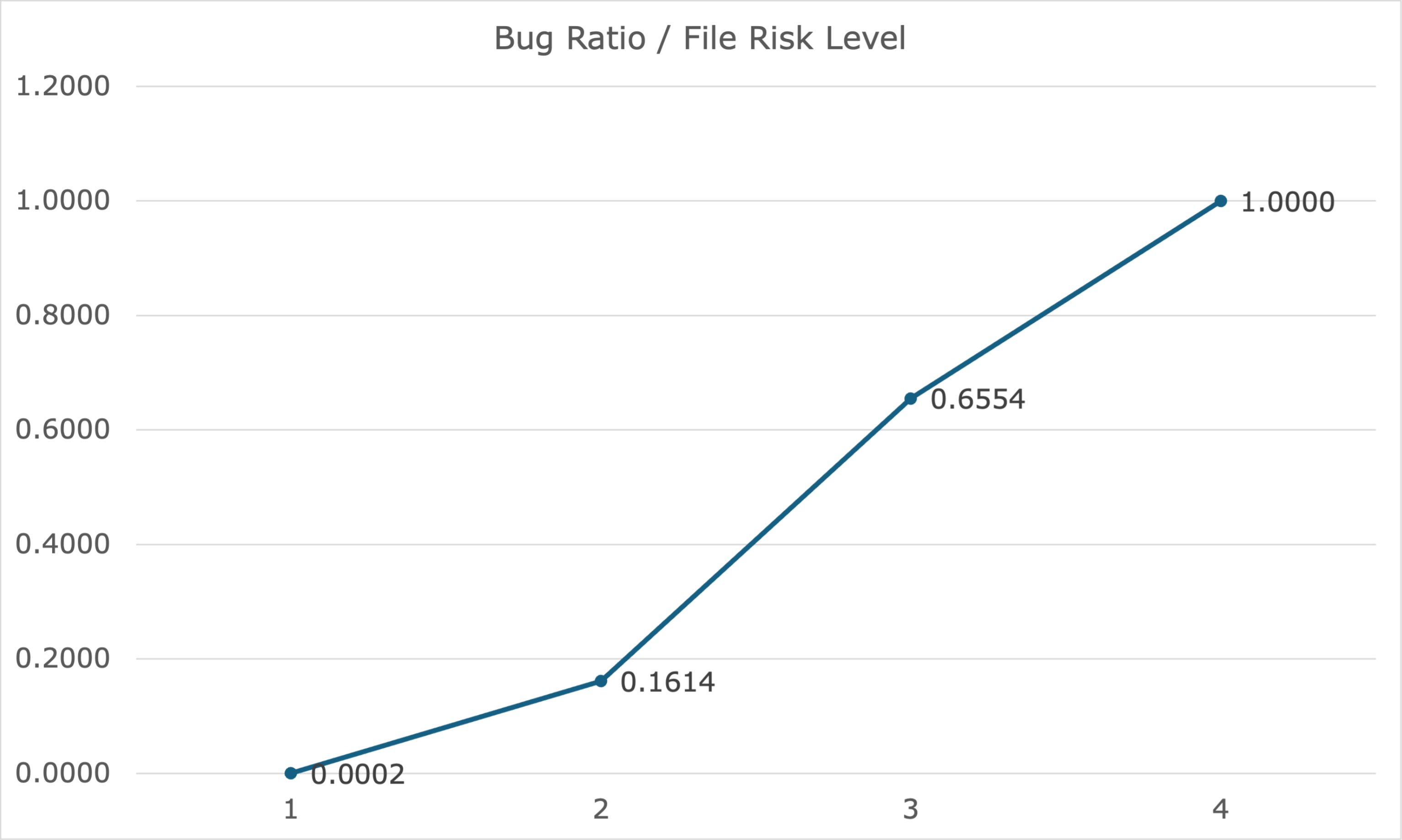


Percentage of code in methods/functions/procedures with more than 15 lines of code	26.81%	< 42%
Percentage of code in methods/functions/procedures with more than 30 lines of code	11.81%	< 19.1%
Percentage of code in methods/functions/procedures with more than 60 lines of code	4.11%	< 6,7%
Unit size risk score:	0	
Percentage of code in methods/functions/procedures with cyclomatic complexity higher than 5	27.74%	< 21.1%
Percentage of code in methods/functions/procedures with cyclomatic complexity higher than 10	12.64%	< 7.6%
Percentage of code in methods/functions/procedures with cyclomatic complexity higher than 25	3.18%	< 0.8%
Total complexity risk:	68	
Percentage of code in methods/functions/procedures with more than 3 argumets	17.59%	< 13.6%
Percentage of code in methods/functions/procedures with more than 5 argumets	4.20%	< 2.8%
Percentage of code in methods/functions/procedures with more than 7 argumets	1.15%	< 0.7%
Total method interfacing risk:	41	

Code risk not zero



Bugs / file and risk level



Risk Level	File Count	Bug Count	Avg Bug Ratio / File	Std. Bugs / File
1	1.048.035	227	0,0002	0,0352
2	316	51	0,1614	0,4806
3	177	116	0,6554	1,2567
4	47	47	1,0000	1,9781

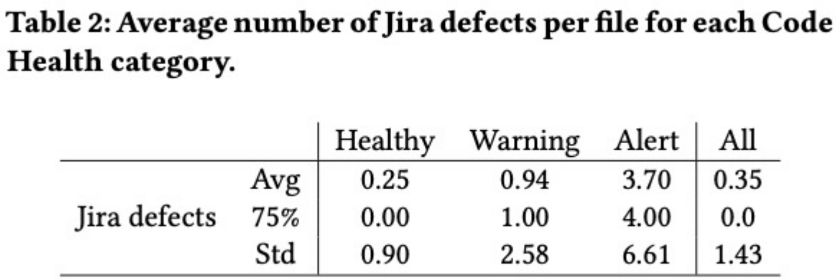
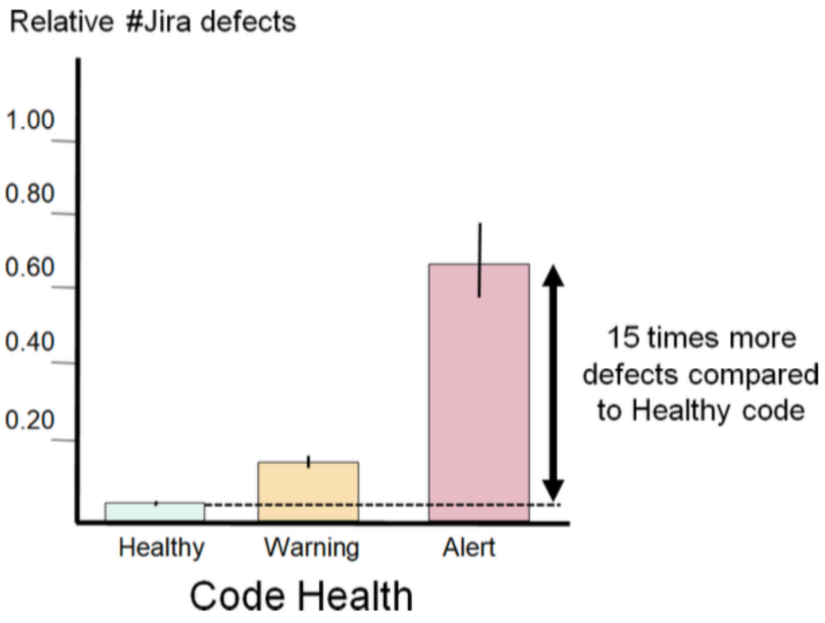
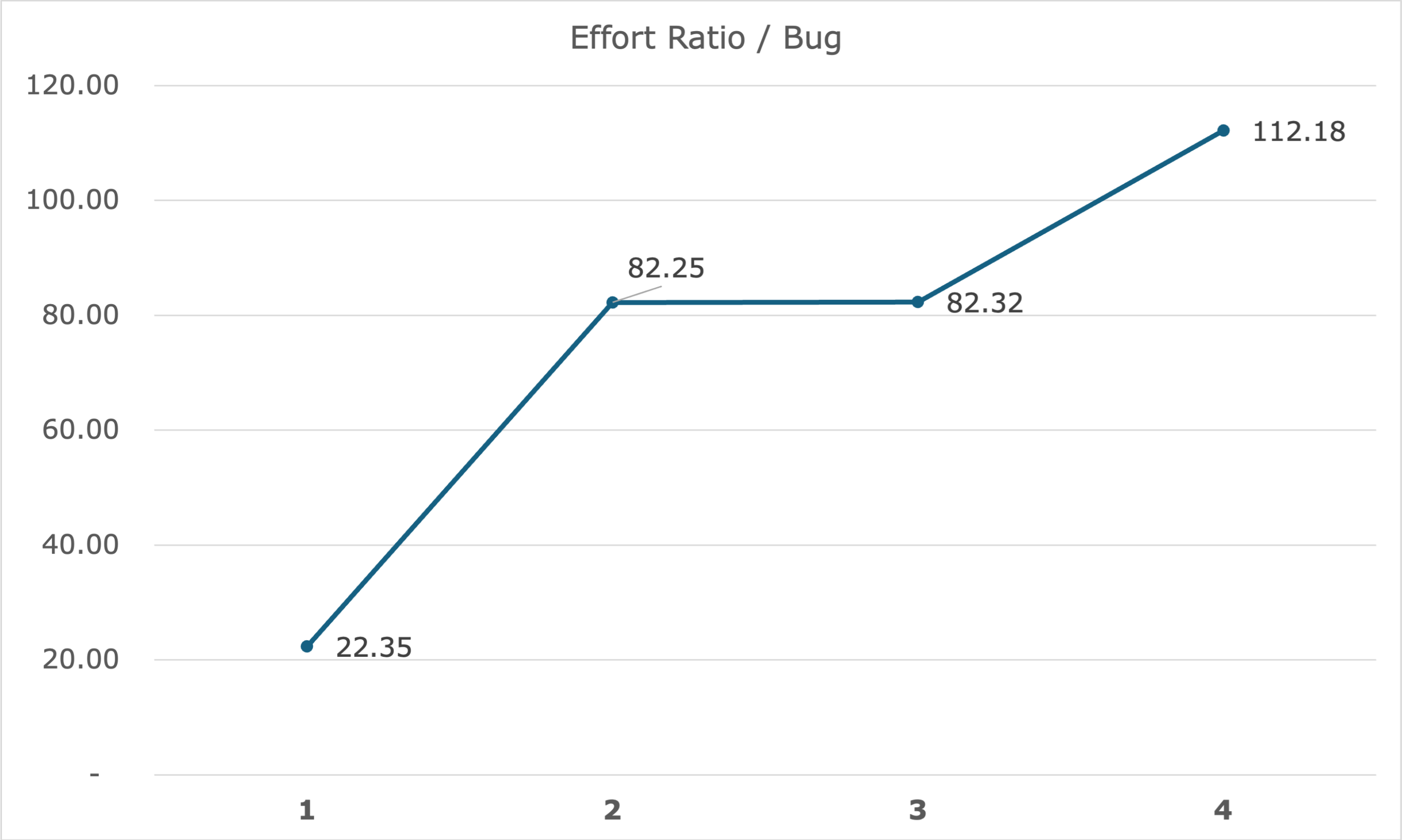


Table 2: Average number of Jira defects per file for each Code Health category.

		Healthy	Warning	Alert	All
Jira defects	Avg	0.25	0.94	3.70	0.35
	75%	0.00	1.00	4.00	0.0
	Std	0.90	2.58	6.61	1.43



Average effort / bug in hours and risk level



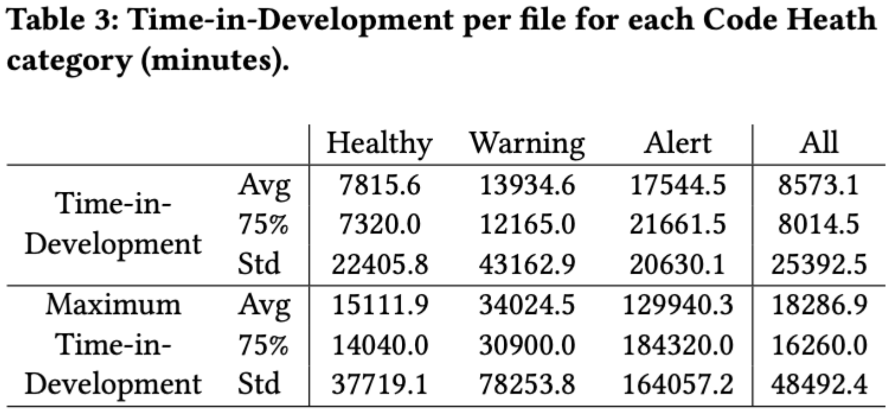


Table 3: Time-in-Development per file for each Code Heath category (minutes).

		Healthy	Warning	Alert	All
Time-in-Development	Avg	7815.6	13934.6	17544.5	8573.1
	75%	7320.0	12165.0	21661.5	8014.5
	Std	22405.8	43162.9	20630.1	25392.5
Maximum Time-in-Development	Avg	15111.9	34024.5	129940.3	18286.9
	75%	14040.0	30900.0	184320.0	16260.0
	Std	37719.1	78253.8	164057.2	48492.4

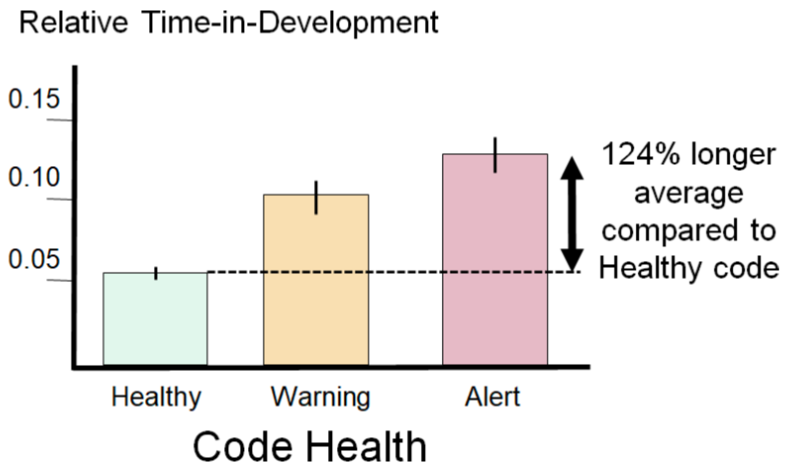
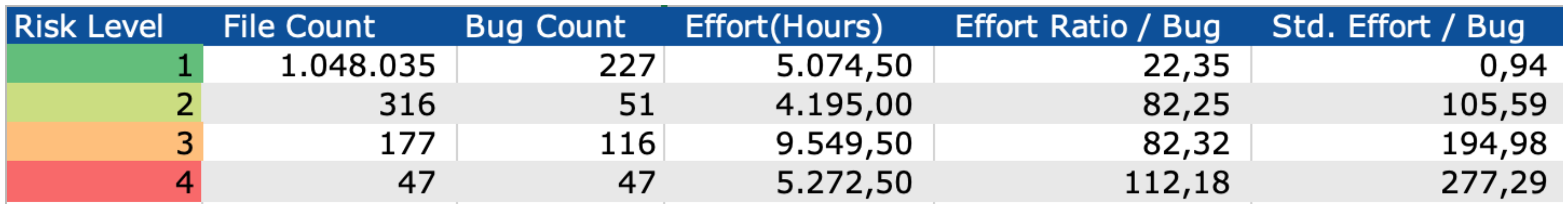
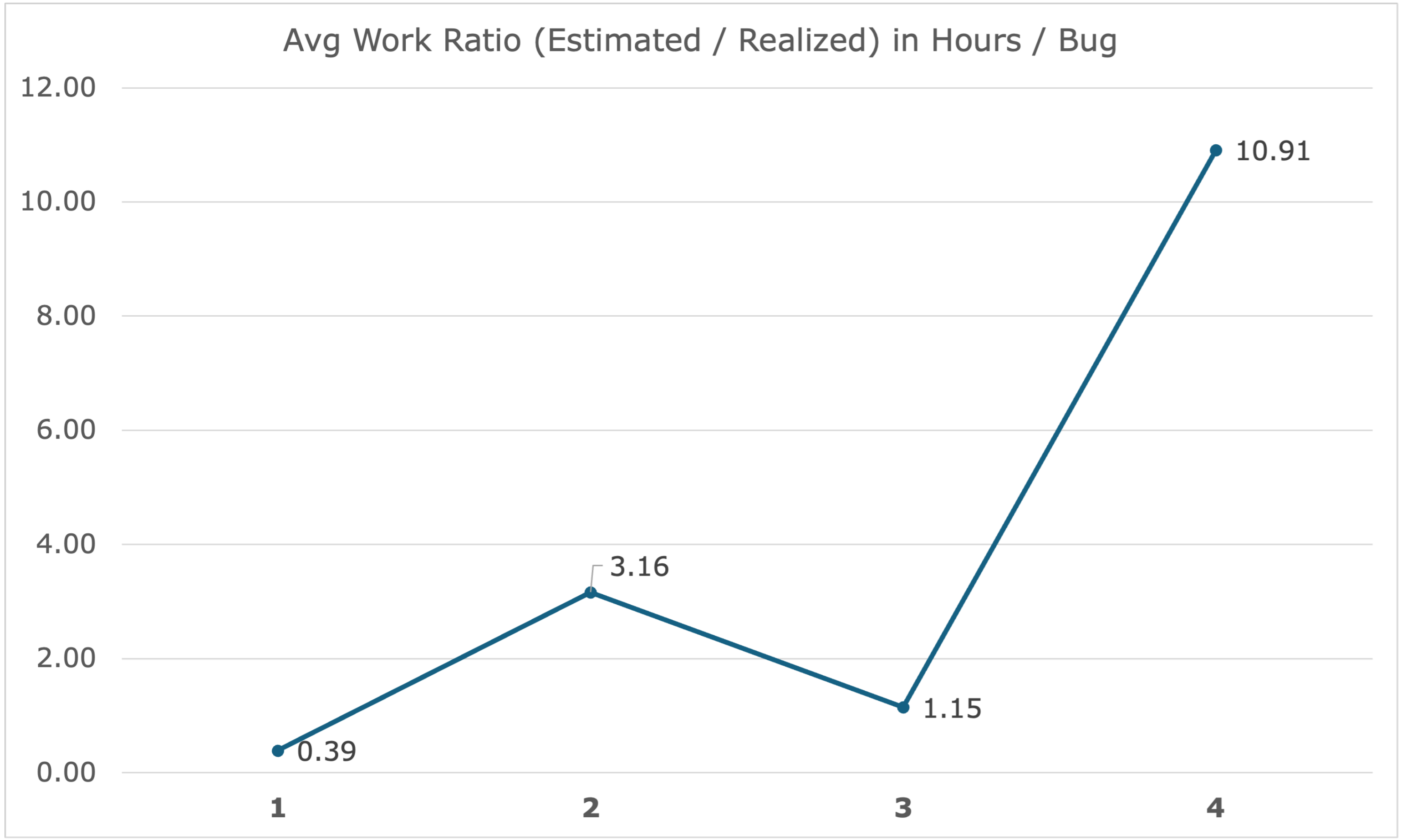


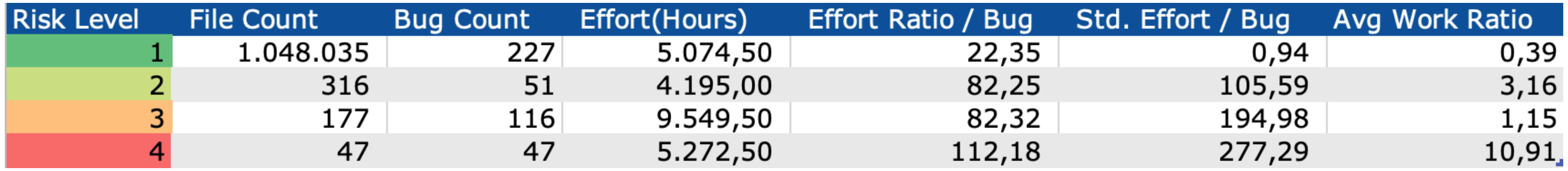
Figure 8: Average Time-in-Development (scaled) for resolving a Jira issue per file. The standard errors are depicted as vertical lines.



Risk Level	File Count	Bug Count	Effort(Hours)	Effort Ratio / Bug	Std. Effort / Bug
1	1.048.035	227	5.074,50	22,35	0,94
2	316	51	4.195,00	82,25	105,59
3	177	116	9.549,50	82,32	194,98
4	47	47	5.272,50	112,18	277,29

Average work ratio (estimated vs realized) / bug and risk level





Risk Level	File Count	Bug Count	Effort(Hours)	Effort Ratio / Bug	Std. Effort / Bug	Avg Work Ratio
1	1.048.035	227	5.074,50	22,35	0,94	0,39
2	316	51	4.195,00	82,25	105,59	3,16
3	177	116	9.549,50	82,32	194,98	1,15
4	47	47	5.272,50	112,18	277,29	10,91

Should we do, something about Technical Debt?

thank you

yonder