



Technical Debt Management in Visma

Mili Orucevic - Chief Software Quality Engineer



Mili Oručević

Chief Software Quality Engineer,
Visma

<https://www.linkedin.com/in/milio>



We are shaping the future of society through technology

17 500

Engaged employees

With more than **6 400** developers

190+ companies make up the Visma Group today

2 100 000
Customers

We are **where** you are
Strong local presence with more than **265+** Locations, **34** markets

Running through Visma's systems every month

11,2 million payslips
26,5 million e-invoices



At Visma the only constant is **change**

Because you rely on our tech
innovations, our world never stands still.

Transform fast and never stop evolving
and innovating

**When writing code, how many of you
have considered the maintainability
aspect?**



You **cannot solve**
technical debt...

...but, you can **manage it!**

...and, not all technical debt is bad...

Lehman's Law of Software Evolution

Law of Continuing Change

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

Law of Increasing Complexity

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

Technical Debt definition

"Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite. Objects make the cost of this transaction tolerable. The **danger occurs when the debt is not repaid**. Every minute spent on **not-quite-right code** counts as **interest on that debt**. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object- oriented or otherwise"

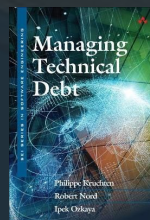
1992 Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)

Ward Cunningham

"In software-intensive systems, technical debt consists of design or implementation constructs that are **expedient in the short term** but that set up a technical context that **can make a future change more costly or impossible**. Technical debt is a contingent liability whose impact is limited to internal system qualities—primarily, but not only, maintainability and evolvability."

Kruchten, Nord, Ozkaya (p.5)

Managing Technical Debt



Software systems are prone to the **build up of cruft** - deficiencies in internal quality that **make it harder than it would ideally be to modify and extend** the system further. Technical Debt is a metaphor, coined by Ward Cunningham, that frames how to think about dealing with this cruft, thinking of it like a financial debt. The extra effort that it takes to add new features is the interest paid on the debt.

Blogpost, 21.05.2019:

<https://martinfowler.com/bliki/TechnicalDebt.html>

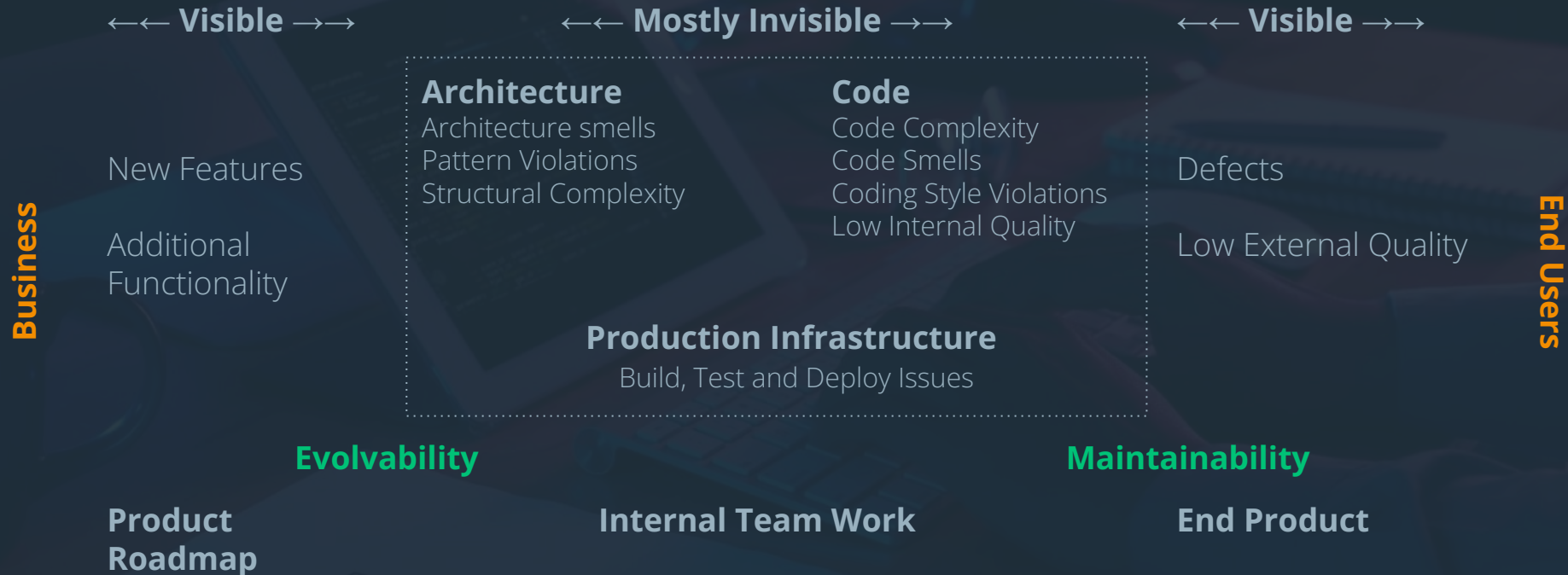
Martin Fowler



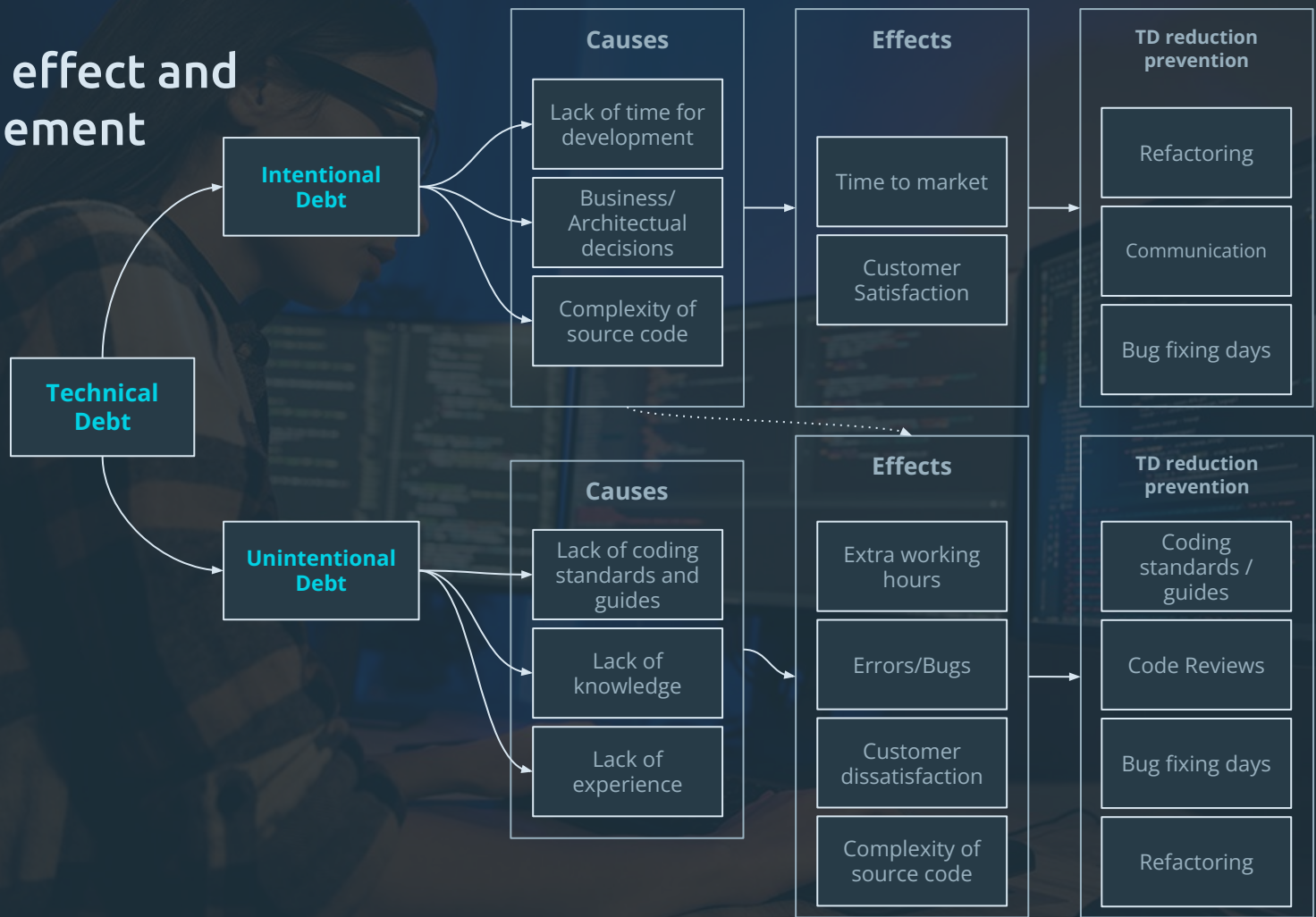
this extra time and effort is the cost of the cruft, paid with each new feature



Technical Debt Landscape



Cause, effect and management





Why should anyone care?



The Verge Menu +

 **Elon Musk** ✓
@elonmusk · Follow

Replying to @pmarca

A small API change had massive ramifications. The code stack is extremely brittle for no good reason.

Will ultimately need a complete rewrite.

7:27 PM · Mar 6, 2023 ⓘ

❤️ 14K 💬 Reply ↗️ Share

[Read 2.3K replies](#)

Despite the company's shrinking workforce, Musk has continued to push for new features, including promising changes that haven't yet appeared, like last month's call for ad revenue sharing with creators who post on the platform or a plan to introduce a new paid tier for its API.

The Verge me engineers Menu +

Some current employees are sympathetic to that view, which places at least part of the blame for Twitter's problems on technical failures that predate Musk's ownership of the company. The fail whale became an icon of the old Twitter for a reason.

"There's so much tech debt from Twitter 1.0 that if you make a change right now, everything breaks," one current employee says.

The Verge vulnerable to catastrophic outages. Menu +

Monday's errant configuration change was at least the sixth high-profile service outage at Twitter this year:

- On January 23rd, Android users temporarily couldn't load new tweets or post them.
- On February 8th, an error message told users that they were "over the daily limit for sending Tweets," preventing them from posting.
- On February 15th, tweets stopped loading.
- On February 18th, the timeline broke and replies disappeared.
- On March 1st, the timeline stopped working.

Rapporten peker på en rekke alvorlige forhold rundt politiets IT-systemer og digitaliseringsarbeid. Rapporten er laget av Deloitte, Footstep og Gnist på oppdrag fra politiet, følge Digi

Kopier

Finn markering

Slå opp



Ingen konkrete planer for utfasing av systemer fra 1990-årene. 80 prosent av IT-budsjettet går til å drifte svært utdaterte løsninger. Politifolk mangler fungerende IT-verktøy, og ledelsen mangler beslutningsgrunnlag.

beslutningsgrunnlag.

fungerende IT-verktøy, og ledelsen mangler

beslutningsgrunnlag.

Det er noen av funnene det pekes på i rapporten.

Politiets IT-budsjett på 1,8 milliarder kroner er det høyeste i staten. Deloitte anslår at det reelle tallet er på 3,2 milliarder kroner, hvis man regner med lønn og administrasjon av IT-feltet. Det meste av budsjettet går til å holde liv i gamle løsninger.

- Rapporten peker på klare svakheter. Derfor jobber vi nå med å etablere en helt ny styringsprosess som integrerer en ny måte å tenke utvikling på i etaten, sier politidirektør Håkon Skulstad til Politiforum.

Riksrevisjonen kritiserte også politiets digitalisering i en rapport i 2023.

i en rapport i 2023.

Riksrevisjonen kritiserte også politiets digitalisering
sier politidirektør Håkon Skulstad til Politiforum.

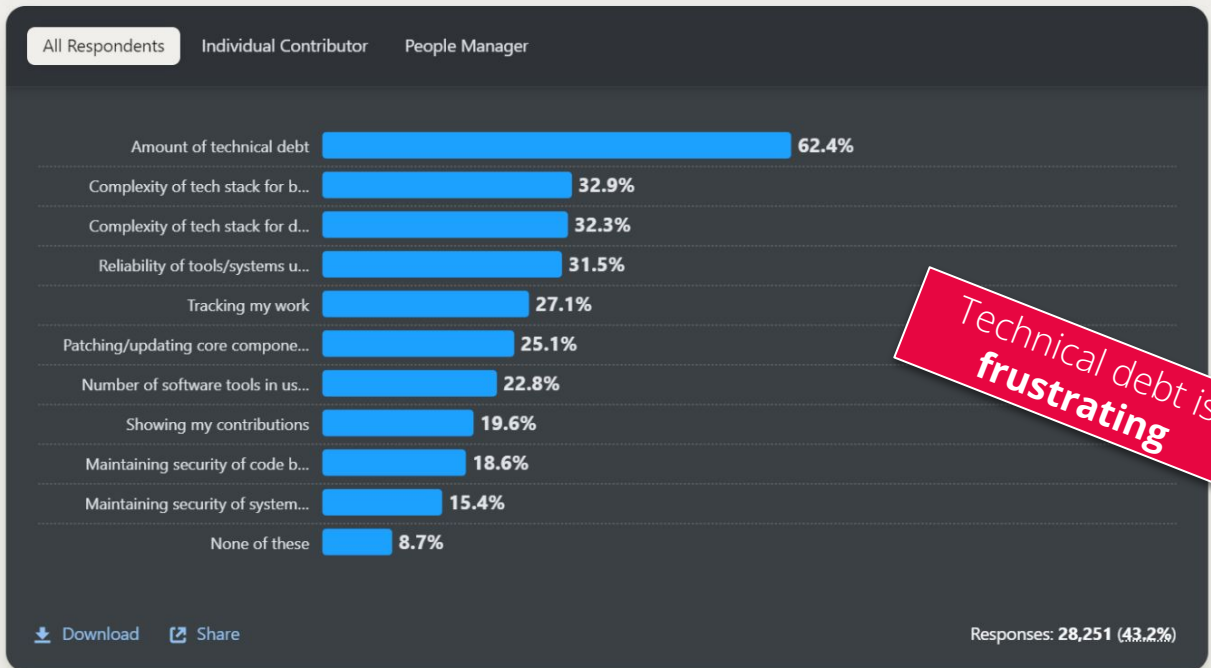
Technical debt causes an **average productivity loss of 36%**
(peaking at 90% in some projects)¹,
costing Visma 3 780 000 hours a year^{2*}

¹ - T. Besker, A. Martini, and J. Bosch, 'Software developer productivity loss due to technical debt—A replication and extension study examining developers' development work', Journal of Systems and Software, vol. 156, pp. 41–61, Oct. 2019, doi: 10.1016/j.jss.2019.06.004.

^{2*} - 36% productivity loss of 6000 developers (2160), 1750 hours a year = 3.78m hours per year

Most common frustrations

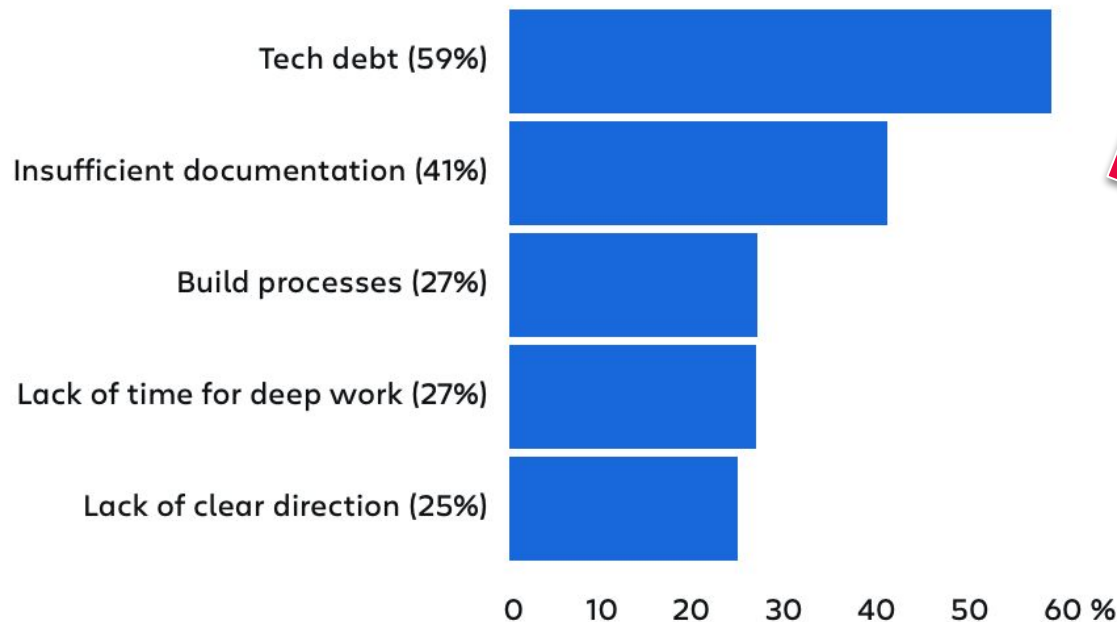
Technical debt is the top frustration at work for professional developers according to 63% of respondents, regardless of whether they are an individual contributor or a people manager.



<https://survey.stackoverflow.co/2024/professional-developers#2-most-common-frustrations>

Top 5 areas of developer time loss according to developers*

*multiple choice question

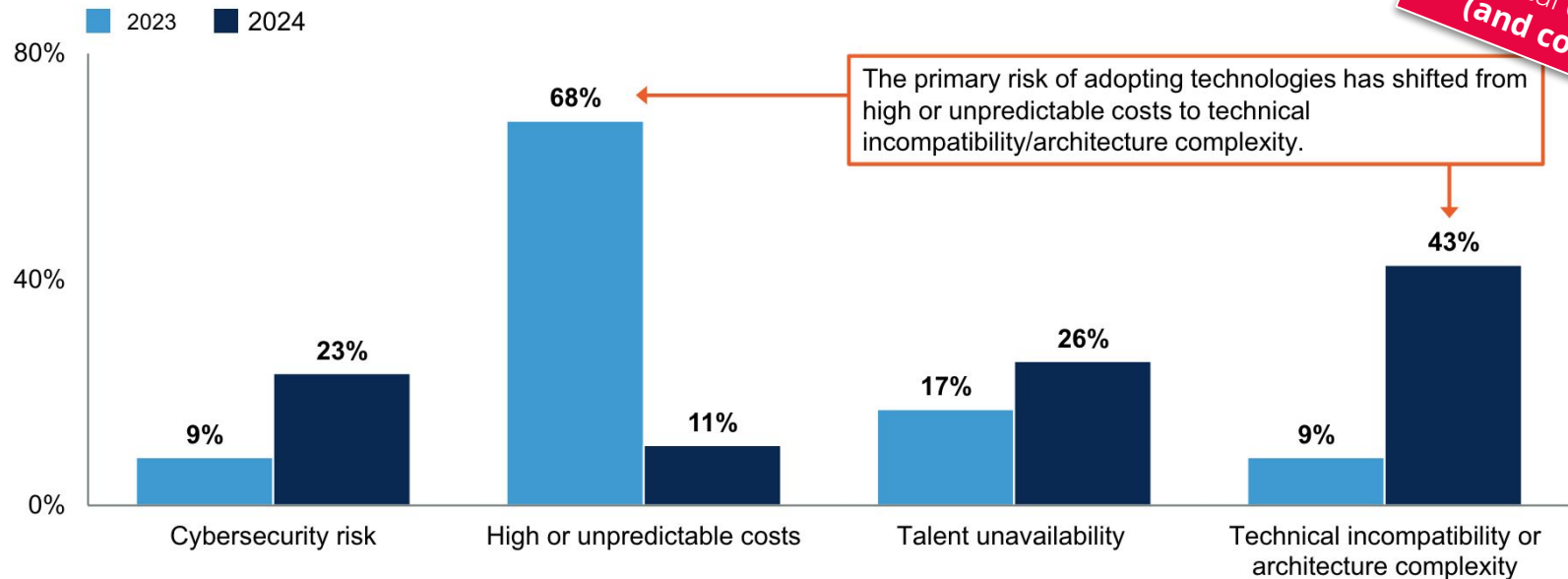


**Technical debt is
time consuming**

<https://getdx.com/report/state-of-developer-experience-report/>

Shift in Risk Perceptions of Technology Adoption

Primary Risk Factors For Software Engineering, 2023 And 2024



n (software engineering leaders from global organizations) = 147; n (technologies) = 47

Source: 2024 Gartner Technology Adoption Roadmap for Software Engineering

© 2024 Gartner, Inc. and/or its affiliates. All rights reserved. 01-2614



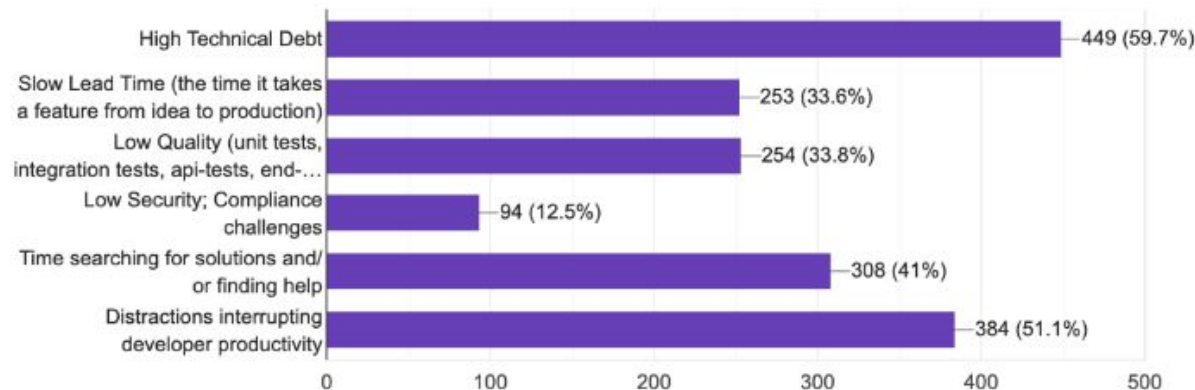
What is negatively affecting your or your engineering team's efficiency and satisfaction?

Amongst other issues affecting productivity, like distractions or searching for solutions, **Technical Debt** and its implications are the biggest pain points for developers and their teams today.

Select which are negatively effecting your or your engineering teams efficiency and satisfaction?

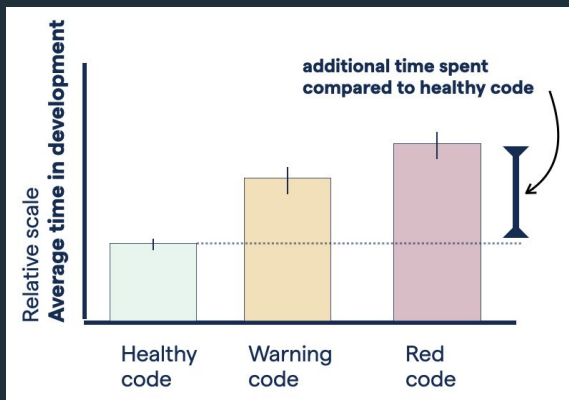


752 responses

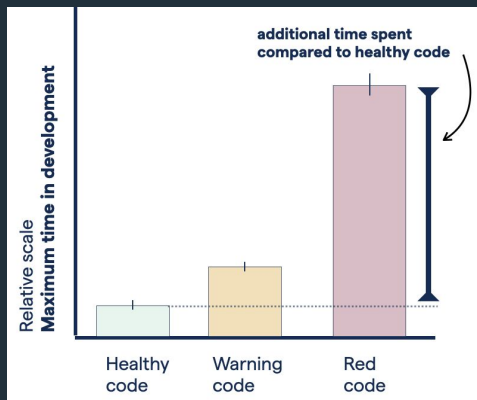


Technical debt is affecting **efficiency** and **satisfaction**!

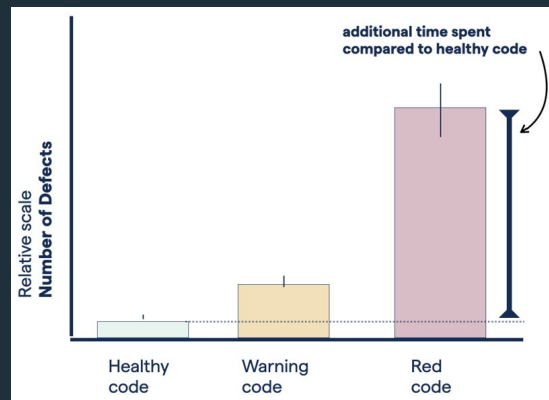
Recent studies from 39 commercial codebases



Implementing a feature or fixing a bug is **twice as expensive** in Red Code



More than **9 times longer** average maximum time leads to uncertainty during development



15 times more defects compared to high-quality code

The aforementioned definition of technical debt implies a careful consideration for each feature that we build. And this is exactly what we do. If we'd build a feature with **0% technical debt, then it would never launch.**

There are **always compromises to be made.**

I think that the question that you have to ask yourself and your team each day is:
Do we have the correct balance between moving forward with some compromises, and adding 'technical debt'?

Moving forward = getting stuff done in a timely manner and launching cool new features that our customers like. **There's always a certain amount of technical debt that's not worth it to iron out.**

- Tech role from Visma

Creating awareness and working with **Technical Debt**

Raise the **awareness** of technical debt

Create **visibility** of technical debt

Common **process** for dealing with technical debt

Keep a technical debt **register**

Ensure that technical debt is **planned** and **repaid** where reasonable





Repayment

How is technical debt repaid?



Prevention

How is introduction of unintentional technical debt prevented?



Planning

How is technical debt included and prioritized in the improvement plans?



Identification

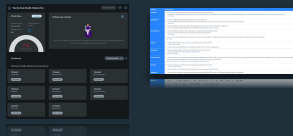
How, and where is technical debt identified?

Technical Debt Assessment



Communication

How is technical debt communicated outside the team, f.ex. towards stakeholders?



Documentation

How is technical debt documented and labeled?



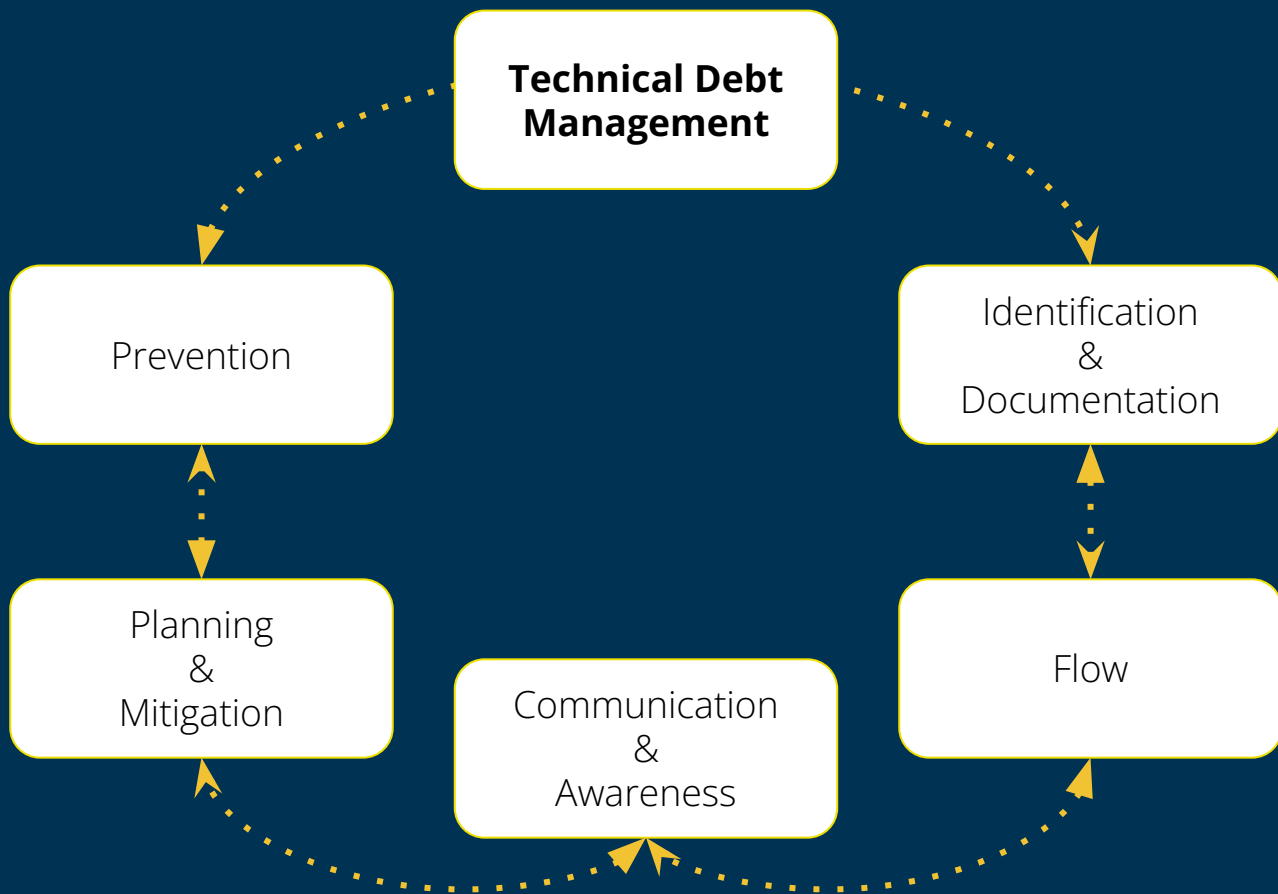
Monitoring

How is technical debt monitored over time, f.ex. trends, top TD issues



Analysis

How is technical debt analyzed with emphasis on risk likelihood, impact and severity?



← Technical Debt Maturity

[Request review](#)

Overview

In progress

Expiration date
Not specified

Open follow-up
actions

Repeats every
365

0



Follow up actions



Your actions points and statuses will be available here once you start your assessment and will link to your tracking system (if access granted).

Contents

Recommended

Technical Debt Maturity Assessment

TDMA01
Prevention

100%

TDMA02
Identification

100%

TDMA03
Documentation

66%

TDMA04
Analysis

100%

TDMA05
Monitoring

Not started

TDMA06
Communication

100%

TDMA07
Planning

50%

TDMA08
Repayment

100%

TDMA01 in progress ▾

Prevention



Prevention capabilities



Q1

The team follows coding standards and performs code review for all non-trivial changes

Open (0) comments

*Answer

Yes

+ Add new action

Q2

The intention of implementing any workaround or corner-cutting is discussed within the team before any code is written; team decides together if the workaround will be implemented or more time will be allocated to avoid it

Open (0) comments

*Answer

Yes

+ Add new action

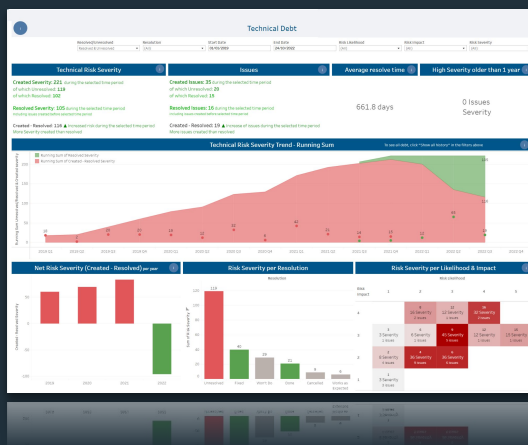
Comments

Send a comment...



No comments yet

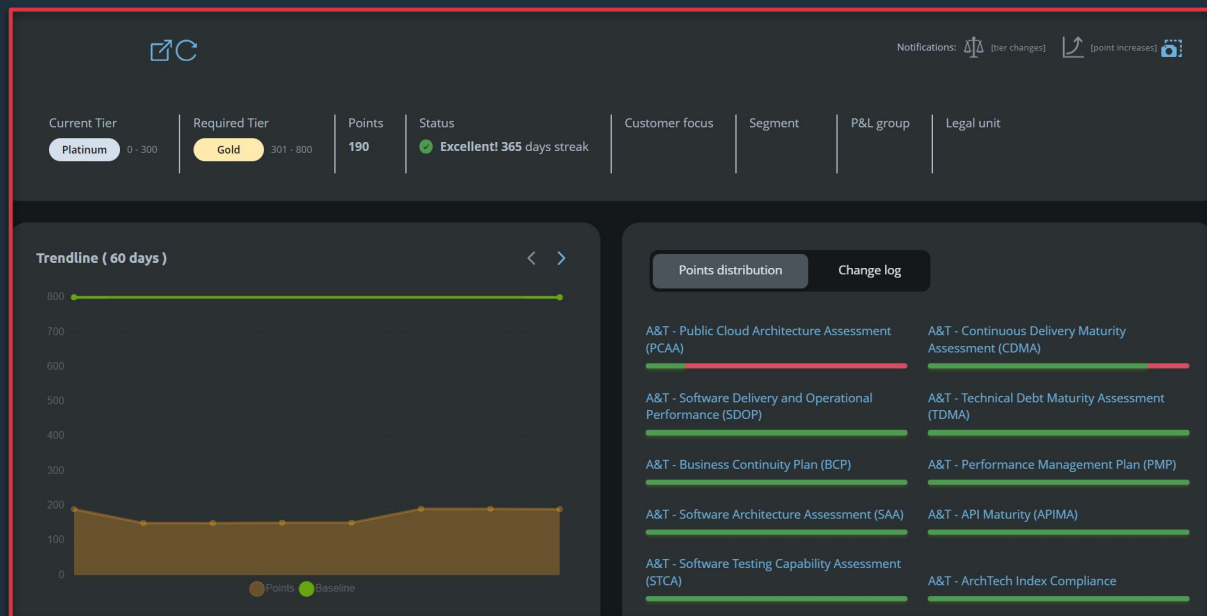
Be the first to comment by typing your message for all to see.



Technical debt **dashboard**

Out of the box

Automatically track deviations



Visma Index

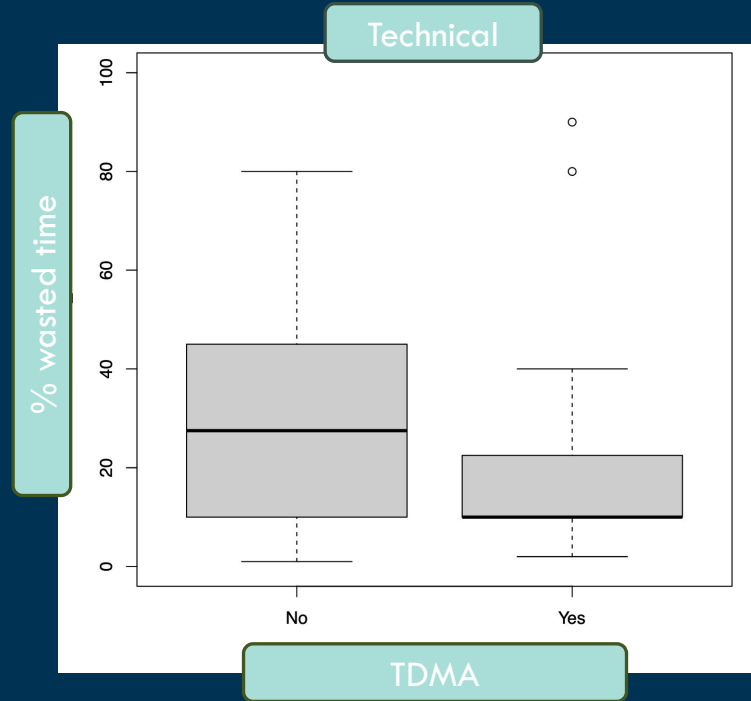
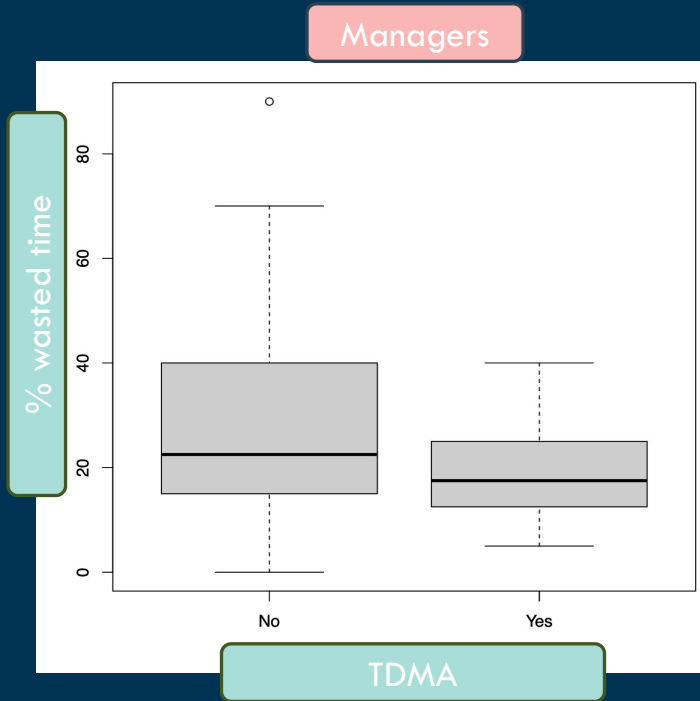
A&T - Technical Debt Maturity Assessment (TDMA)

1 . Technical Debt Maturity Assessment not done or report older than 12 months	Hubble/Confluence 	0	1	0	
2 . Technical Debt Maturity Assessment not done or report older than 18 months	Hubble/Confluence Due date in 6 months 	0	450	0	
3 . Unresolved critical and blocker issues from the Tech Debt Assessment older than 30 days	Jira  Total: 0 	0	200	0	
4 . Unresolved high issues from the Tech Debt Assessment older than 90 days	Jira  Total: 0 	0	100	0	
5 . Unresolved medium issues from the Tech Debt Assessment older than 90 days	Jira  Total: 0 	0	20	0	
6 . Unresolved low and minor issues from the Tech Debt Assessment older than 90 days	Jira  Total: 0 	0	1	0	
7 . Data Quality - Open technical debt cases, older than 14 days, missing severity score	Jira 	4	0	0	
8 . Data Quality - Closed technical debt cases without severity score (closed last 12 months)	Jira 	3	0	0	
9 . Data Quality - Open technical debt cases with severity score, missing NFR label	Jira 	0	0	0	
10 . Data Quality - Closed technical debt with severity score but missing NFR label	Jira 	2	0	0	
11 . 0 technical debt cases created last 3 months	Jira  Total: 3 	0	0	0	
12 . Less than 5 technical debt cases created last 12 months	Jira  Total: 20 	0	0	0	
13 . 0 technical debt cases resolved last 3 months	Jira  Total: 4 	0	0	0	
14 . Less than 5 technical debt cases resolved last 12 months	Jira  Total: 14 	0	0	0	
				0	



The screenshot displays a Jira dashboard with multiple sections. At the top, there are filters for 'All' and 'Technical Debt'. Below this, there are several charts and tables. The main table shows a list of technical debt items with columns for 'Item', 'Status', 'Severity', 'Due Date', and 'Assignee'. The items are grouped by 'Category' and 'Status'. The dashboard also includes a 'Summary' section with a table of counts for different categories and statuses. The overall layout is clean and organized, providing a comprehensive overview of technical debt management.

Technical Debt Assessment is associated with Less Wasted Time



- **"Just Ship It" Hardcoded Configuration**
- **The Outdated Dependency**
- **Skip the tests**
- **Knowledge loss**
- **Multiple technologies**



AI & Technical Debt

AI-introduced Dependency Bloat

AI-generated Black Box

Well defined small, focused issues - AI can help here

Why can not AI just solve these issues?

- Usually not a coding decision but a business decision with a lot of context involved
- AI does not know the entire context or the intent
- Technical debt is not about fixing one function, it is often understanding components across the entire system, which is not possible to load into AI's memory (although some possibilities exist)
- AI's training data - trained on massive data amounts from the net, which is a mix of brilliant, clean code and abandoned messy and outdated code - it will generate what is statistically probably, not always the “best” solution. It often suggest older libraries, or suggest a popular-but-flawed pattern, and trade in a technical debt for another one.

1 Bad Line of Code == 1 Bad Line of Code

1 Bad Line of Plan == 10-100 Bad Lines of Code

Wrong Solution

1 Bad Line of Research == 1000+ Bad Lines of Code

Misunderstanding the System

1 Bad Line of Specification == 10000+ Bad Lines of Code

Wrong Problem

1 Bad Line of Command/CLAUDE.md == 100000+ Bad Lines of Code

Core Infrastructure

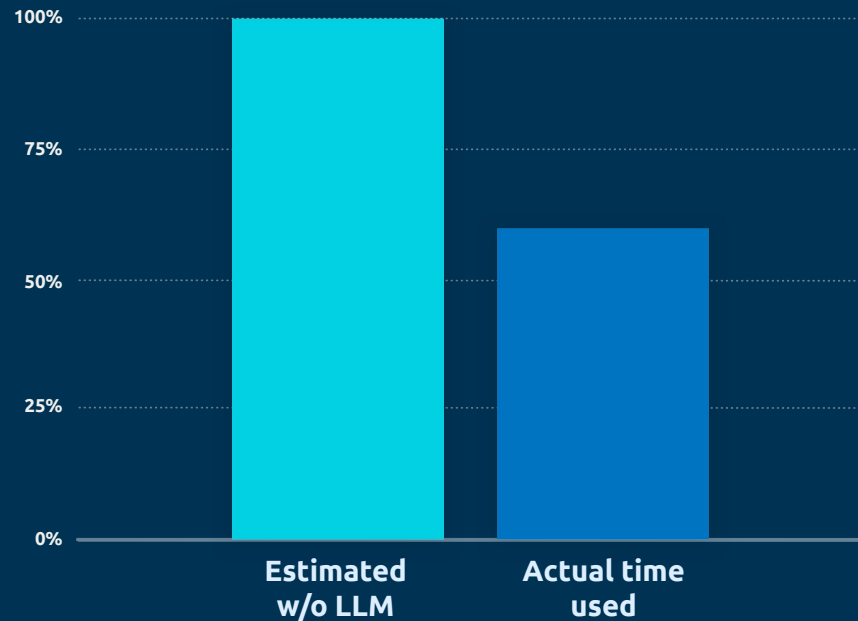
Workshop: Modernise large codebases using GenAI

Team / Challenge

- Migrate 2M+ lines of code from .Net Fw 4.8 to .Net 8
- Tightly coupled monolithic architecture, Windows OS
- Evaluate effectiveness of GenAI

Acceleration

- Project estimation ~20k hours (w/o LLM)
- Estimated time saving when using LLMs ~40%
- Hosting cost savings: ~500k EUR / year



Innovation Project for the Industrial Sector

Data-driven continuous management of technical debt for sustainable software development

SINTEF, University of Oslo, Akva Group, KnowIT and Visma



Data-driven continuous management of technical debt for sustainable software development

- 3 years
- 15.2MNOK funding from Norwegian Research Council
- Visma, AkvaGroup, Knowit, UiO, SINTEF

Research

- Novel **data-driven** methods and software tools aimed at aiding companies to **control technical debt systematically and continuously**
- Methods and software tools to improve current software engineering practices and validate the novel approaches, **saving million of wasted working hours yearly**, for many years to come



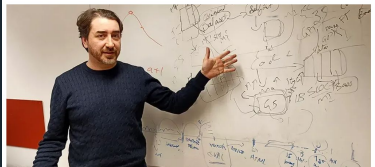
Research

- **Knowledge sharing**, webinars to raise awareness
- **Guest lectures** from University of Groningen, Rey Juan Carlos University of Madrid, University of São Paulo
- Presentations **from companies in UiO, NTNU**
- State of technical debt survey and **perception of technical debt** and how this is perceived and develops over 3 years
- Automated **labeling of technical debt** from project backlog
- Automated **classification** of technical debt
- Automated **prioritization** of technical debt
- **Business Impact of Technical Debt**
- debtGuardian - scan code repository for potential **security and technical debt** and suggest fixes **using GenAI**
- Exploring Arcan, Designite, DV8, CodeScene,- a tool focusing on **identifying architectural debt** → **large-scale study on TD analysis tools**
- Experimenting with using **LLMs for refactoring**
- Unify findings from different tools into **one signal**
- **Visualization** of Technical Debt
- **Prompting strategies** for better technical debt detection
- Technical debt in **AI generated code**
- ...



Brukar maskinl ring for betre handtering av IT-etterslep

I mange IT-prosjekt oppst r behov for betydelig etterarbeid, av ulike  rsaker. Det vert kalla teknisk gjeld. No h per forskarane   behandle gjelda mer effektivt med maskinl ring.



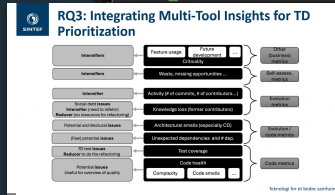
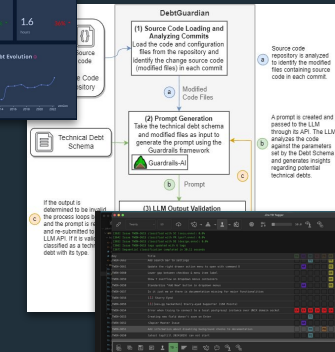
Language/Framework/Technology	Version	Layer	Primary or Other significant	Risk
TypeScript	5.5.x	Backend	Primary	High
JavaScript	ES2023	Backend	Primary	High
React	18.3	Frontend	Primary	High
TypeScript	5.7.x	Frontend	Primary	High

TABLE 1
SUMMARY OF INDIVIDUAL PROJECTS USED IN THIS STUDY

Product	Short description	Programming languages	Rank Composites	LOC	Tool used	# References
A	A modern software platform	C# .Net 6, TypeScript, Android	25	120K	CodeScene, Designite	3
B	A modern software platform supporting data integration	C# .Net 6, Android, Python	76	100K	Arcan, CodeScene, DV8	3
C	A legacy system with a lot of legacy code, some parts are not easily maintainable	C#, C++, Python	10	2.00K	Arcan, CodeScene	3
D	A legacy system with a lot of legacy code, some parts are not easily maintainable	JavaScript, C#, TypeScript	35	1.00K	Arcan, CodeScene	3

The discussed architectural smell:

1) CodeScene: For each .NET project (product analyzed), we select and examine aggregated metrics in detail. CodeScene's code health metric provides a refactoring-oriented view (see Table 1B). We managed to have CodeScene-based analysis sessions with the product teams for the software products A, B, C. Even though their average code health indices are high (>90%), yet issues point to high-activity files, or hotspots, primarily within "Real Code". Hotspots indicate areas of frequent development where technical debt may hinder progress. CodeScene prioritizes refactoring recommendations by subunit files based on low code health



Entrepreneurial Responsible Dedicated Inclusive

Champions of business software

