

Technical Debt Management in Visma

Maren Maritsdatter Kruke
Security Business Analyst

Mili Orucevic
Chief Software Quality Engineer



languages

h, Swift,

possible

la, Dart,

gian, C#,

ish, C++,

C Sharp,

l, Kotlin,

Finnish.



**covering
mission
critical
processes...**



We simplify and automate complex processes

So you can grow your
business and satisfy laws
and regulations



Visma in numbers

15 800

Engaged employees
with nearly **6 000**
developers

1 900 000

Customers

€ 2 392m

Value created for
society in 2023

€ 2 719m

Revenue LTM 3Q2024

€ 882m

EBITDA LTM 3Q2024

190+

Companies make up the
Visma Group today

11,3m & 25,7m

payslips

e-invoices

running through Visma's systems every
month

265+

Locations - strong local
presence
We are where you are

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

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 +

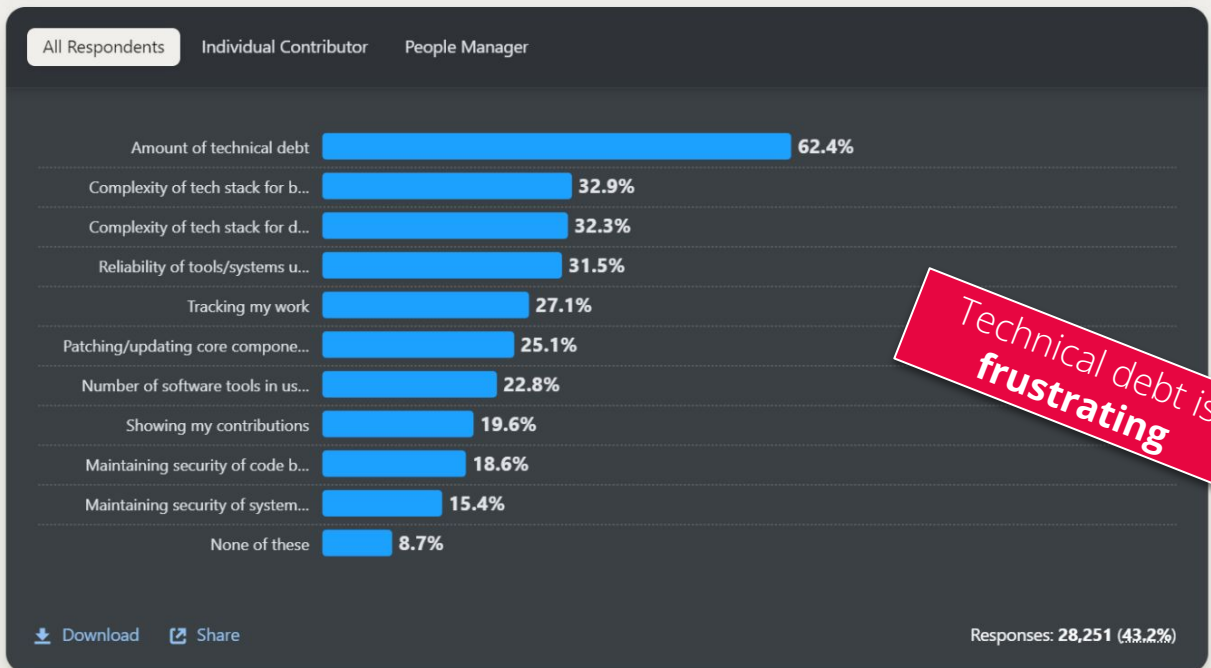
Monday's errant configuration change was at least the sixth high-profile service outage at Twitter this year:

- 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.

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.

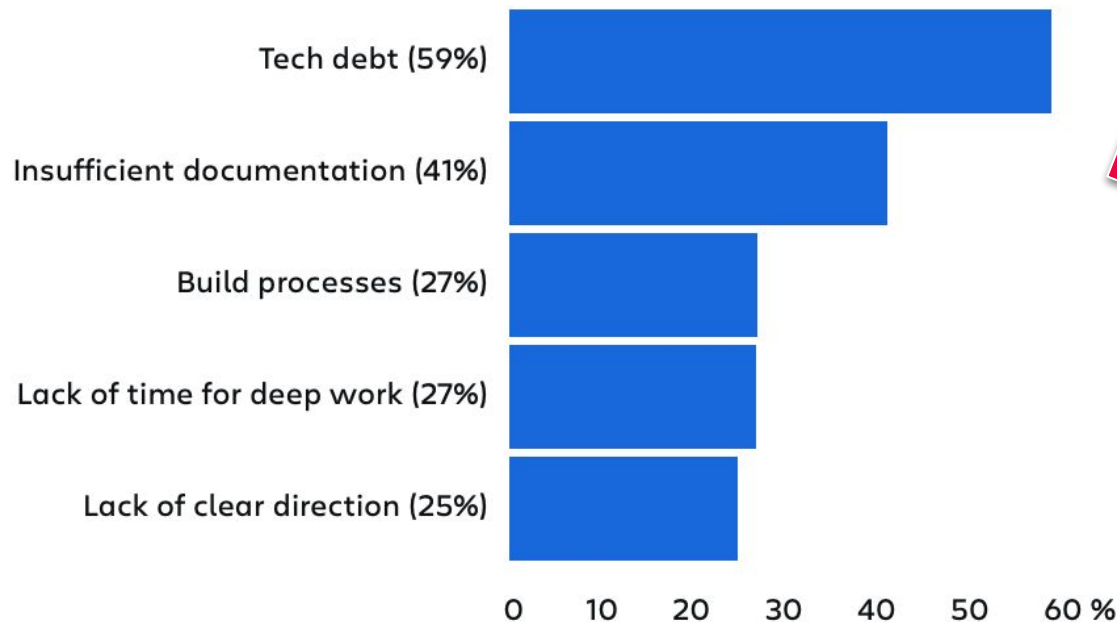
? Which of these company challenges causes you the most frustration? Select all that apply.



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

Top 5 areas of developer time loss according to developers*

*multiple choice question

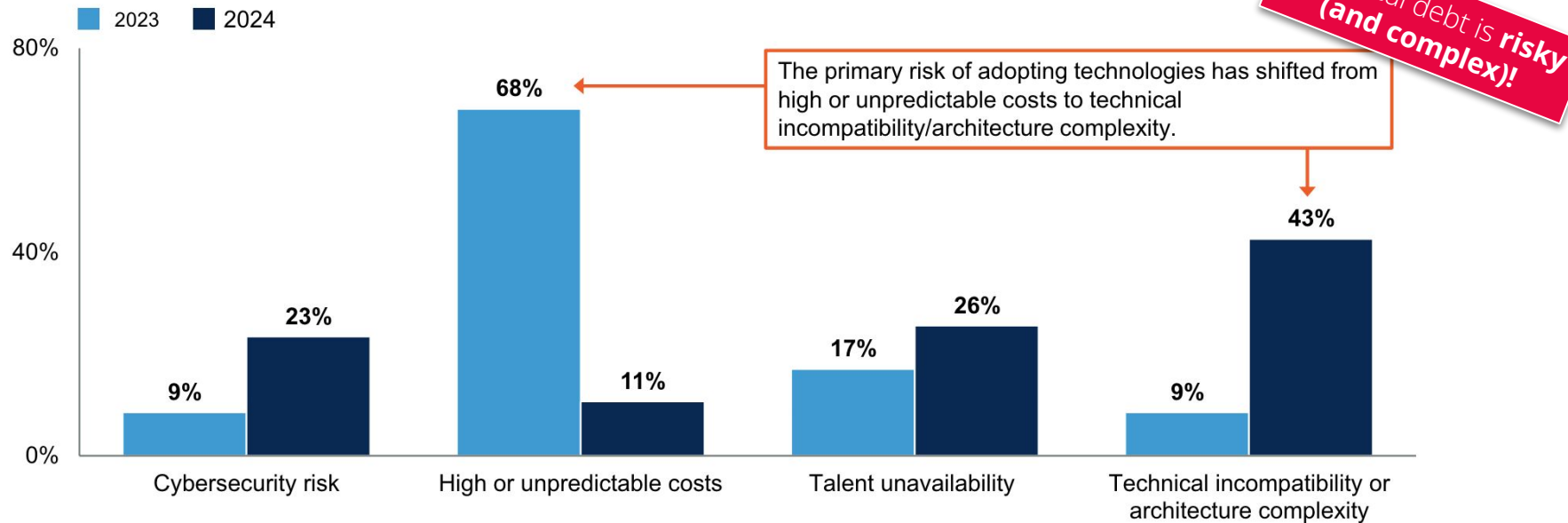


**Technical debt is
time consuming**

<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



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



How do we
manage technical debt
at Visma?

Creating awareness and working with **Technical Debt**

Raise the **awareness**
of technical debt in
organizations

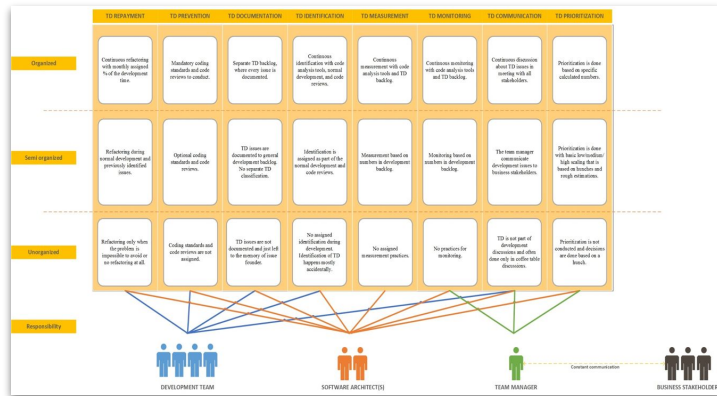
Create **visibility** of
technical debt

A **process** for dealing
with technical debt

Keep a technical debt
register

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





Inspired by research done at the
Lappeenranta University of Technology
by Jesse Yli-Huumo.

Technical Debt Self-assessment Maturity Model

TDM activity / TDM levels	TD Repayment	TD Prevention	TD Documentation	TD Identification	TD Measurement	TD Monitoring	TD Communication	TD Prioritization
Organized (level 3)	☒ Continuous refactoring with monthly assigned % of the development time.	☒ Mandatory coding standards and code reviews to conduct.	Same as Semi organized level	Continuous identification with code analysis tools, normal development, and code reviews. Identification as part of problem management and operational processes.	☒ Continuous measurement with code analysis tools and TD backlog.	☒ Continuous monitoring in technical health measurement tool (e.g. SonarQube) with code analysis tools and TD backlog. ☒ Monitoring show both qualitative (TD backlog etc) and quantitative measurements (static measurements)	Continuous communication about TD issues in meeting with all stakeholders.	☒ Prioritization is done based on specific calculated numbers. Recommended to use Severity as the calculated number
Semi organized (level 2)	☒ Refactoring during normal development and previously identified issues.	☒ Optional coding standards and code reviews.	☒ TD issues are documented to general development backlog. No separate TD classification.	☒ Identification is assigned as part of the normal development and code reviews. ☒ Completed self assessments.	☒ Measurement based on numbers in development backlog.	☒ Monitoring based on numbers in development backlog.	☒ Product/Service Owner communicate development issues to business stakeholders. Communication must describe customer value.	☒ Prioritization is done with basic low/medium/high scaling that is based on hunches and rough estimations.
Unorganized (level 1)	Refactoring only when the problem is impossible to avoid or no refactoring at all.	Coding standards and code reviews are not assigned.	TD issues are not documented and just left to the memory of issue founder.	No assigned identification during development. Identification of TD happens mostly accidentally.	No assigned measurement practices.	No practices for monitoring.	TD is not part of development discussions and often done only in coffee table discussions.	Prioritization is not conducted and decisions are done based on a hunch.
Responsibility for activity	Development team, software architect(s)	Development team, software architect(s)	Development team, software architect(s)	Development team, software architect(s)	Software architect(s)	Software architect(s), team manager, product/service owner	Development team, software architect(s), team manager, product/service owner	Software architect(s), team manager, product/service owner

Technical Debt Maturity

Request review

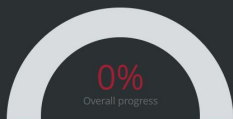


Overview

In progress

Expiration date
Not specified
Repeats every
365

Open follow-up
actions
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

Not started

TDMA02
Identification

Not started

TDMA03
Documentation

Not started

TDMA04
Analysis

Not started

TDMA05
Monitoring

Not started

TDMA06
Communication

Not started

TDMA07
Planning

Not started

TDMA08
Repayment

Not started

Technical Debt Maturity

Task 1 of 8



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

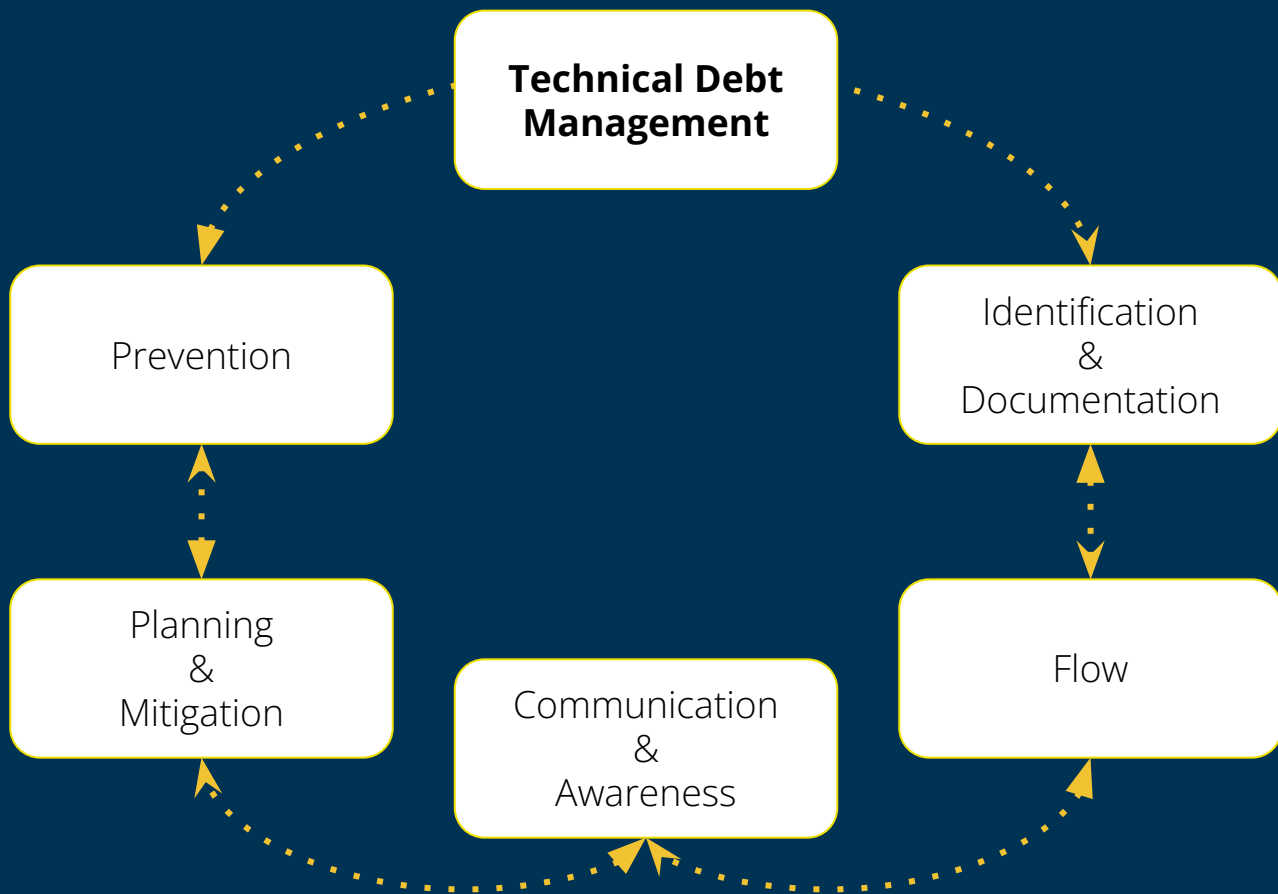
Comments

Send a comment...



No comments yet

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



← Technical Debt 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

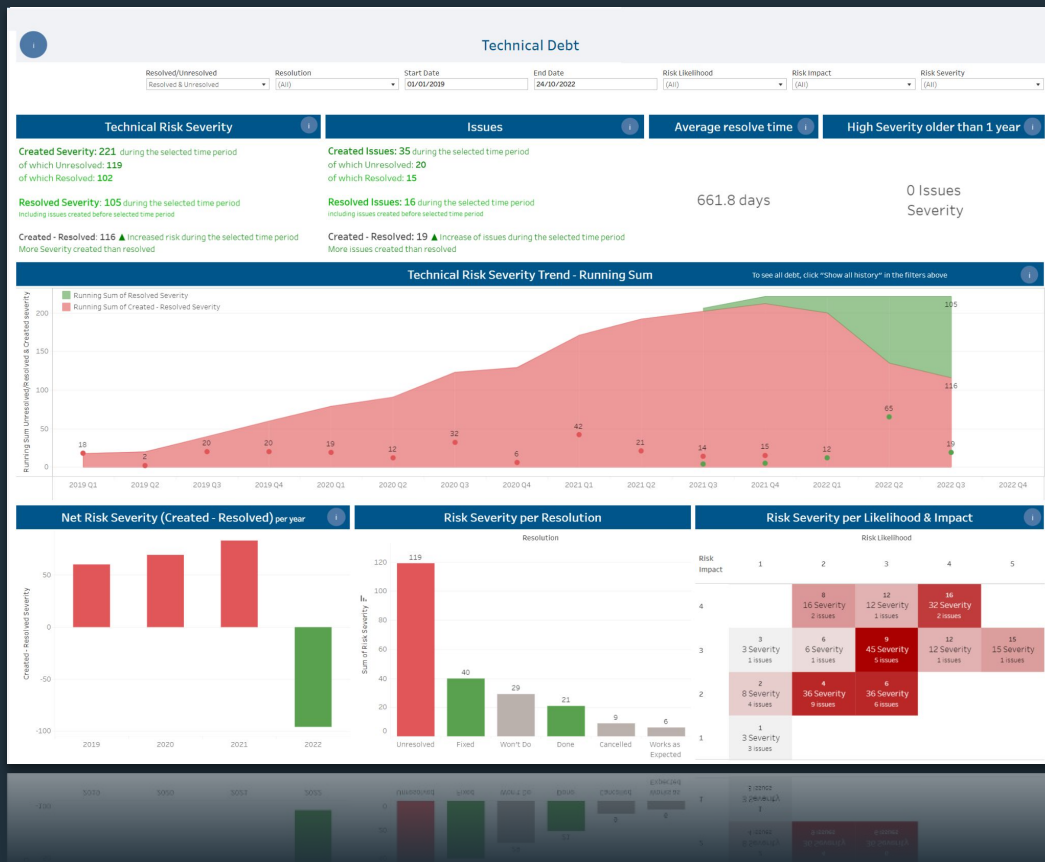
100%

TDMA07
Planning

50%

TDMA08
Repayment

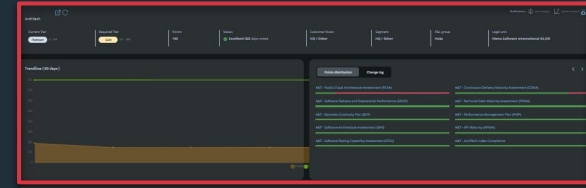
100%



Technical debt dashboard

Out of the box

Visma Index

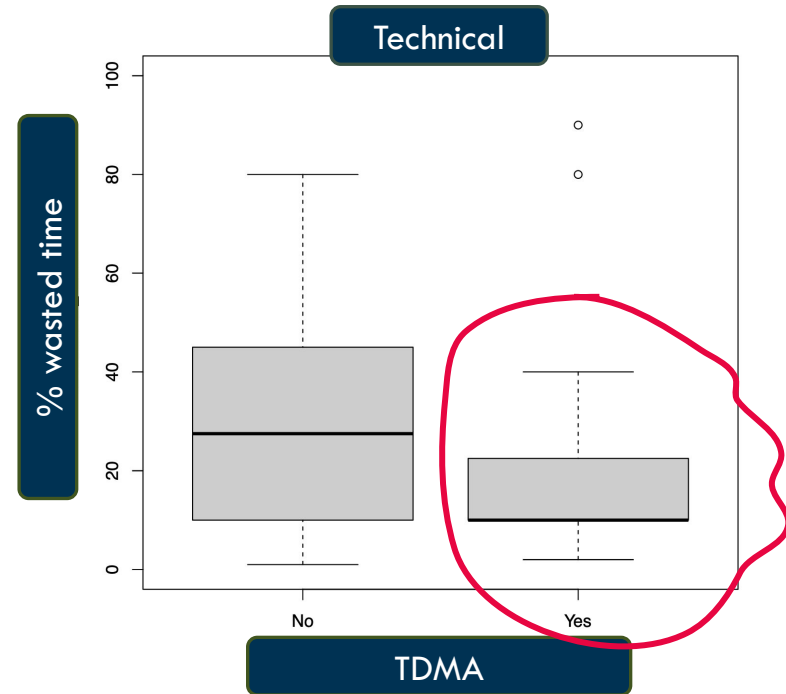
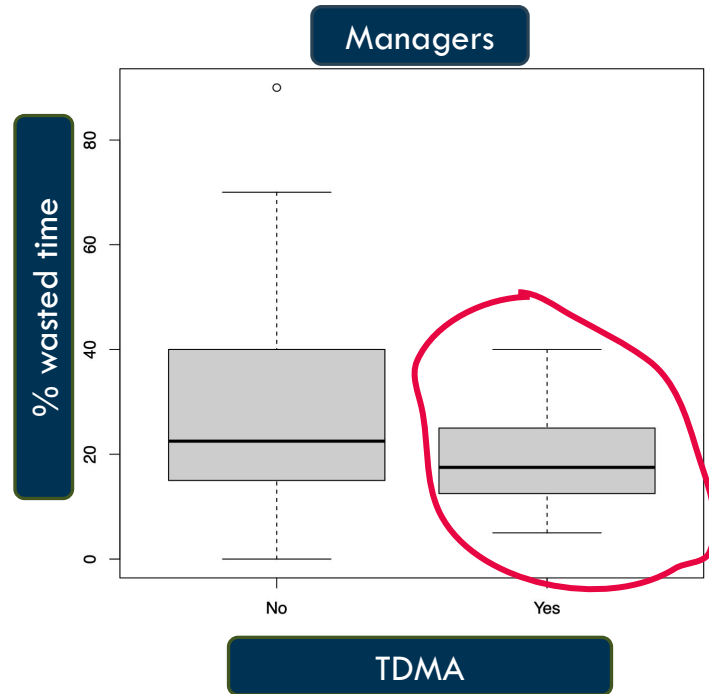


Visma Index

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	

“Teams who have done the TDMA **waste less time**”



Examples from SonarQube

QUALITY GATE STATUS

Passed
All conditions passed

MEASURES

New Code
Since November 10, 2020

Overall Code

QUALITY GATE STATUS

Failed
1 conditions failed

MEASURES

New Code
On New Code

Overall Code

Failed
1 conditions failed

On New Code

1 New Critical Issues is greater than 0

MEASURES

New Code
Since 1.0.217-
SNAPSHOT
Started 6 hours ago

Overall Code

Conditions

Conditions on New Code
Conditions on New Code apply to all branches and to Pull Requests.

Metric	Operator	Value
Coverage	is less than	60.0%
Duplicated Lines (%)	is greater than	3.0%

Conditions on Overall Code
Conditions on Overall Code apply to branches only.

Metric	Operator	Value
Coverage	is less than	80.0%
Duplicated Lines (%)	is greater than	2.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Rating	is worse than	A
Unit Test Errors	is greater than	0
Unit Test Failures	is greater than	0

11 Bugs

4 Vulnerabilities

0 Security Hotspots

13d Debt

340 Code Smells Maintainability **A**

77.7%
Coverage on 17K Lines to cover

994
Unit Tests

2.0%
Duplications on 41k Lines

53
Duplicated Blocks

Examples from SonarQube

The screenshot displays the SonarQube web interface. On the left, a sidebar lists several 'Code Smell' issues, all categorized as 'Minor'. The main area shows a code editor with a C# snippet. A pink tooltip is visible over the code, stating: 'Remove the unused local variable 'request'. Why is this an issue?'. The tooltip also indicates the issue is 'unused', has a '5min effort', and includes a 'Comment' link. The code in the editor is as follows:

```
67 }
68
69 private async Task<[redacted]> ([redacted] Name)
70 {
71     [redacted]
72     var request = new GetSecretValueRequest { SecretId = secretName};
73 }
```

Unused local variables should be removed

Unused local variables should be removed

Code Smell Minor unused Available Since Aug 14, 2015 SonarQube (C#) Constant/Issue: 5min

If a local variable is declared but not used, it is dead code and should be removed. Doing so will improve maintainability because developers will not wonder what the variable is used for.

Noncompliant Code Example

```
public int NumberOfMinutes(int hours)
{
    int seconds = 0; // seconds is never used
    return hours * 60;
}
```

General exceptions should never be thrown

Utility classes should not



Examples from SonarQube

The screenshot displays the SonarQube web interface. At the top, there is a navigation bar with tabs: Overview, Issues, Security Hotspots, Security Reports, Measures, Code, and Activity. The 'Issues' tab is selected. On the right side of the navigation bar, there is a 'Project Information' link. Below the navigation bar, the main content area is titled 'Complete the task associated to this 'TODO' comment.' and includes a 'Get permalink' link. A sub-header 'Track uses of "TODO" tags' is followed by filters: 'Code Smell' (selected), 'Info', 'Open', and 'Not assigned'. Below these filters are two tabs: 'Where is the issue?' (selected) and 'Why is this an issue?'. The main content area shows a code editor with a file named 'y.cs'. The code includes a public interface 'CreateLinksFor' and a method 'Guard.ArgNotNull'. A TODO comment is present: '// TODO: cover with tests think of smarter solution here (https://...)'. A red box highlights the TODO comment with the text 'Complete the task associated to this 'TODO' comment.' and a 'Code Smell' icon. On the left side of the interface, there is a sidebar with a search bar and a list of issues. The first issue in the list is 'Complete the task associated to this 'TODO' comment.' with a 'Code Smell' icon.

Overview Issues Security Hotspots Security Reports Measures Code Activity

Project Information

Get permalink

Code Smell Info Open Not assigned

Where is the issue? Why is this an issue?

See all issues in this file

```
37 /// <inheritdoc />
38 public ICalendar CreateLinksFor(ICalendar obj)
39 {
40     Guard.ArgNotNull(obj, nameof(obj));
41
42     // TODO: cover with tests think of smarter solution here (https://...)
43
44
45
46
47
48
49
50
51
```

Complete the task associated to this 'TODO' comment.

Code Smell

“Not our fault”

Example of a bigger technical debt

- Unintentional
- External factors
- Time

“Move from AngularJS to new version of Angular”

~ 1583 hours = 66 days



Security debt **definition**

“Security debt is a set of design or implementation solutions that hinder or has the **potential** to hinder the achievement of a system’s security goal”

Security debt **management**

- **Identification**

- Bug Bounty
- Security testing performed by a security team in Visma

- **Prevention**

- Threat modelling

} **Specific to SD management**

- **Approaches used for TD management -> can be used for SD management**

- Prevention - coding standards
- Prevention & Identification - code review
- Identification - tools

- **Security knowledge is **important****

Security debt prioritization/planning

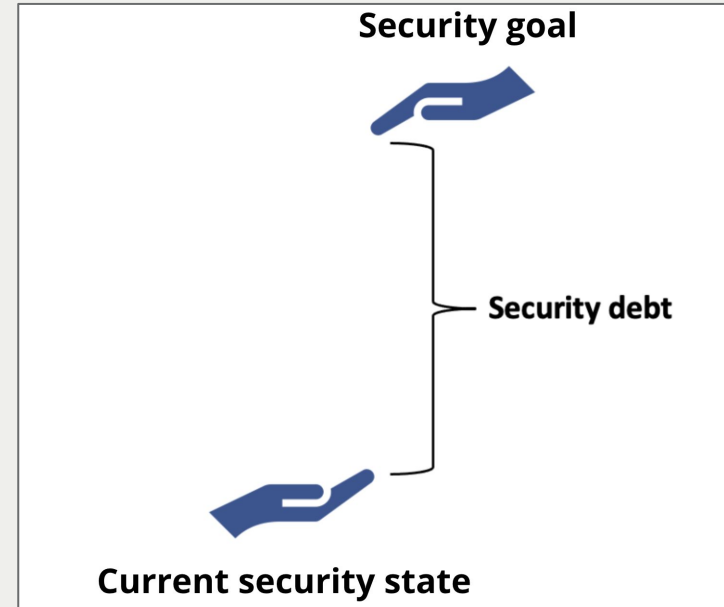
- **Security debt and technical debt **balance****
- **Key points for security debt prioritization**
 - Communication
 - Labels
 - Severity / priority
- **Labels to identify security debt**
 - "security"
 - "<technical_debt>"
- **Compare values for security debt prioritization**



Security debt accumulation patterns

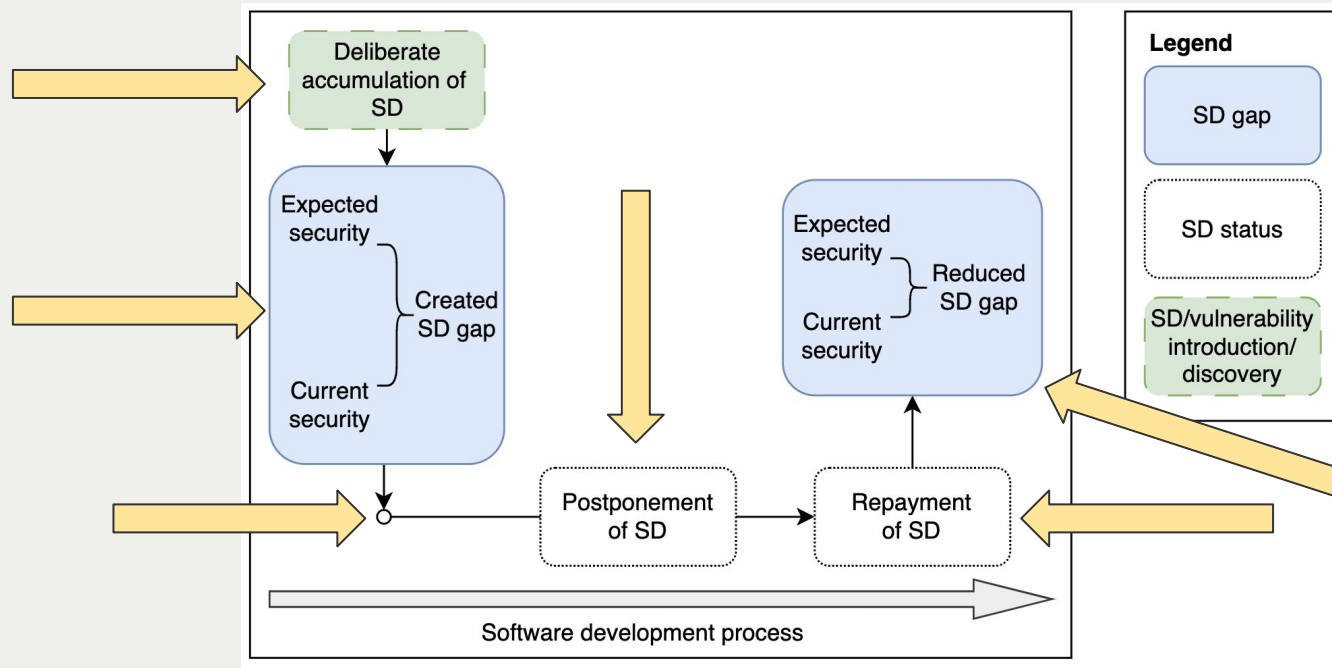
Overview

- Keeping security debt **under control**
- **Patterns:**
 - Deliberate
 - Non deliberate
 - Security vulnerabilities
 - Potential security risk
- **Common denominators in the patterns:**
 - Security debt exist in the system while being **postponed**
 - When repayment is performed, the security debt **gap is reduced**



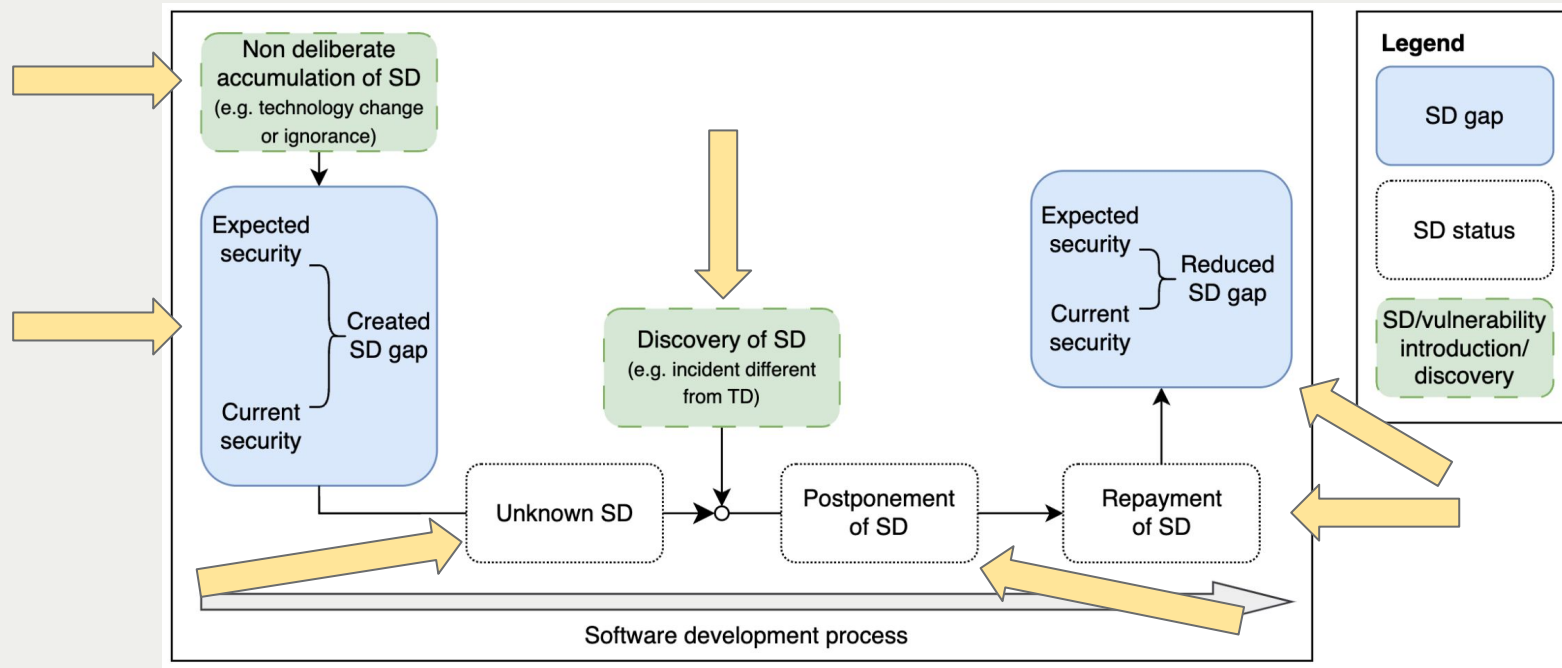
Security debt accumulation patterns

Deliberate security debt (SD)



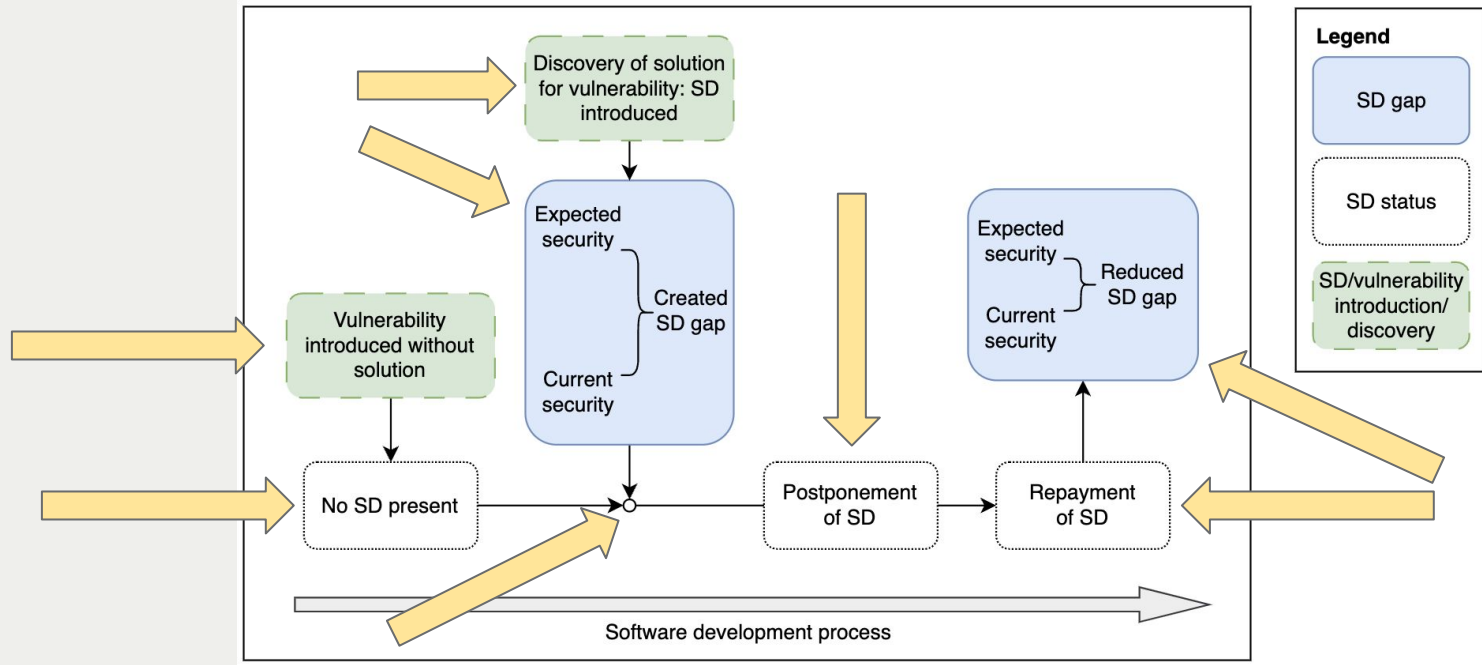
Security debt accumulation patterns

Non deliberate security debt (SD)



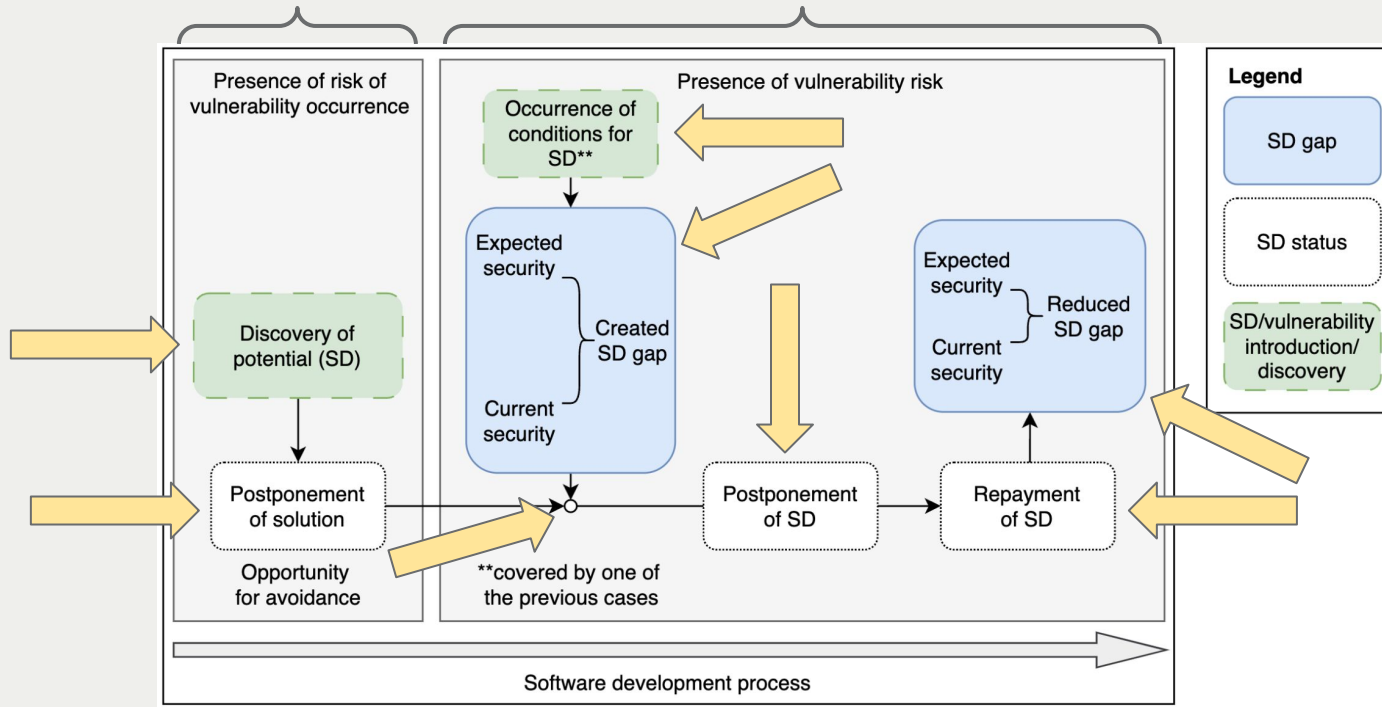
Security debt accumulation patterns

Security vulnerabilities as security debt (SD)



Security debt accumulation patterns

Security debt (SD) with potential security risk



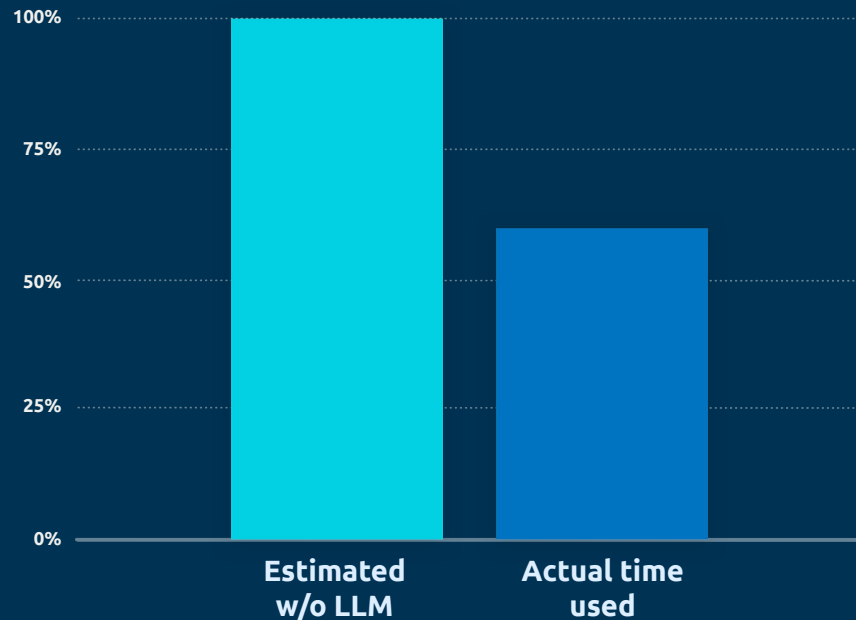


What about using AI to
fix Technical Debt?

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



Found means fixed!



github-advanced-security bot

...

The vulnerability in the code is due to the fact that user-provided input is directly used in HTTP response without any sanitization. This can lead to a cross-site scripting (XSS) attack if the user input contains malicious scripts.

To fix this, we need to sanitize the input before using it in the HTTP response. One way to do this is to use the **escape-html** library, which can escape any special characters.

AI suggested fix

```
14     app.get('/', (req, res) => {  
15         -   res.send('Hello ${req.query.name}!');  
15         +   res.send('Hello ${escape(req.query.name)}!');  
16     });
```

<https://github.blog/news-insights/product-news/found-means-fixed-introducing-code-scanning-autofix-powered-by-github-copilot-and-codeql/>

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

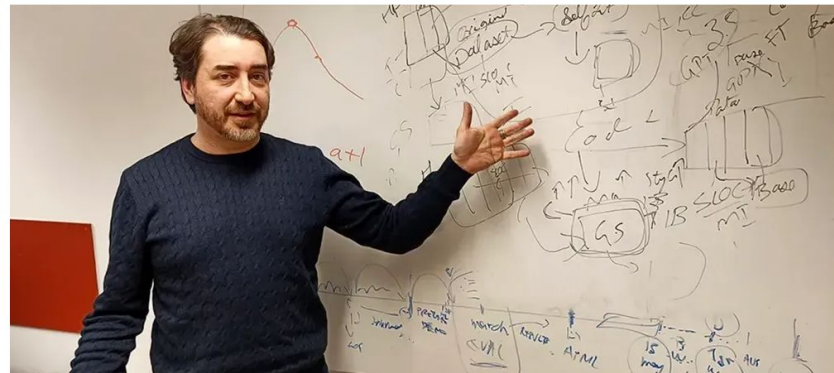
- **Detection** of technical debt issues in backlog
- Automated **labeling**
- Risk **assessment** - type of td, priority
- Experimenting with using **LLM for refactoring**
- DebtGuardian - using LLMs to find potential technical and security **debt in pull requests**

Read more on:

<https://www.digi.no/artikler/brukar-maskinlaering-f-or-betre-handtering-av-it-etterslep/543204>

Brukar maskinlæring for betre handtering av IT-etterslep

I mange IT-prosjekt oppstår behov for betydeleg etterarbeid, av ulike årsaker. Det vert kalla teknisk gjeld. No håper forskarane å behandla gjelda meir effektivt med maskinlæring.



Entrepreneurial Responsible Dedicated Inclusive

Champions of business software

