Technical Debt Management in Visma

Mili Orucevic
Chief Software Quality Engineer

VISMA


JUMBO
VISMA
HEMA

We are shaping the future of society through technology

14 500
Engaged employees

With more than **5 500** developers

300 companies
have joined Visma the last decade

1 600 000
Customers

We are where you are
Strong local presence with more than **265+** locations

Running through Visma's systems every month

11,2 million payslips
22,2 million e-invoices

VISMA

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."

VISMA

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"

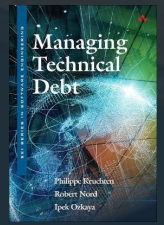
1992 Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)

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



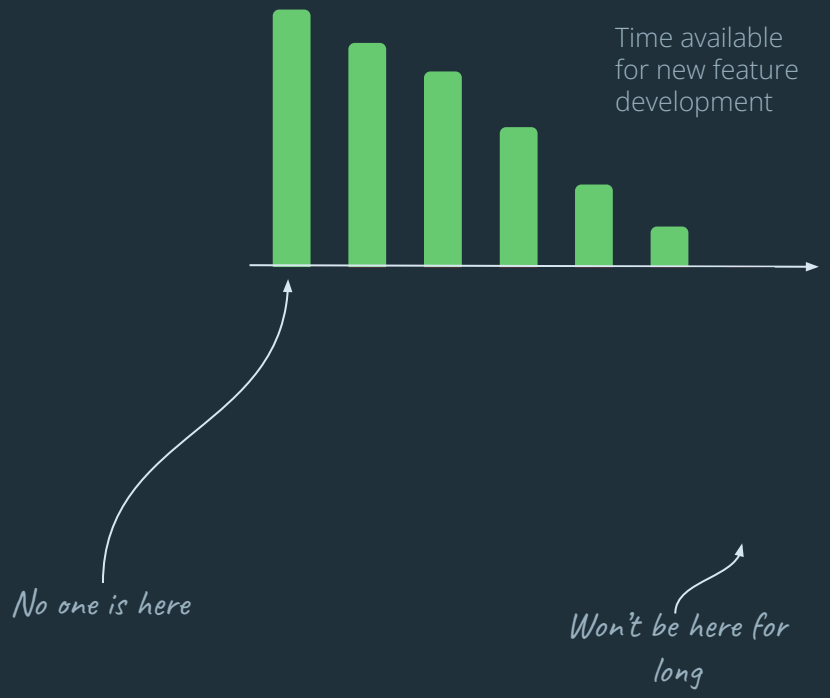
Software systems are prone to the **build up of cruft** - deficiencies in internal quality that **make it harder than it would ideally be to modify and extend** the system further. Technical Debt is a metaphor, coined by Ward Cunningham, that frames how to think about dealing with this cruft, thinking 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

VISMA

Visualizing Technical Debt



VISMA

Technical Debt Landscape

←← Visible →→

←← Mostly Invisible →→

←← Visible →→

Business

New Features

Additional Functionality

Architecture
Architecture smells
Pattern Violations
Structural Complexity

Code
Code Complexity
Code Smells
Coding Style Violations
Low Internal Quality

Production Infrastructure
Build, Test and Deploy Issues

Defects

Low External Quality

End Users

Evolvability

Maintainability

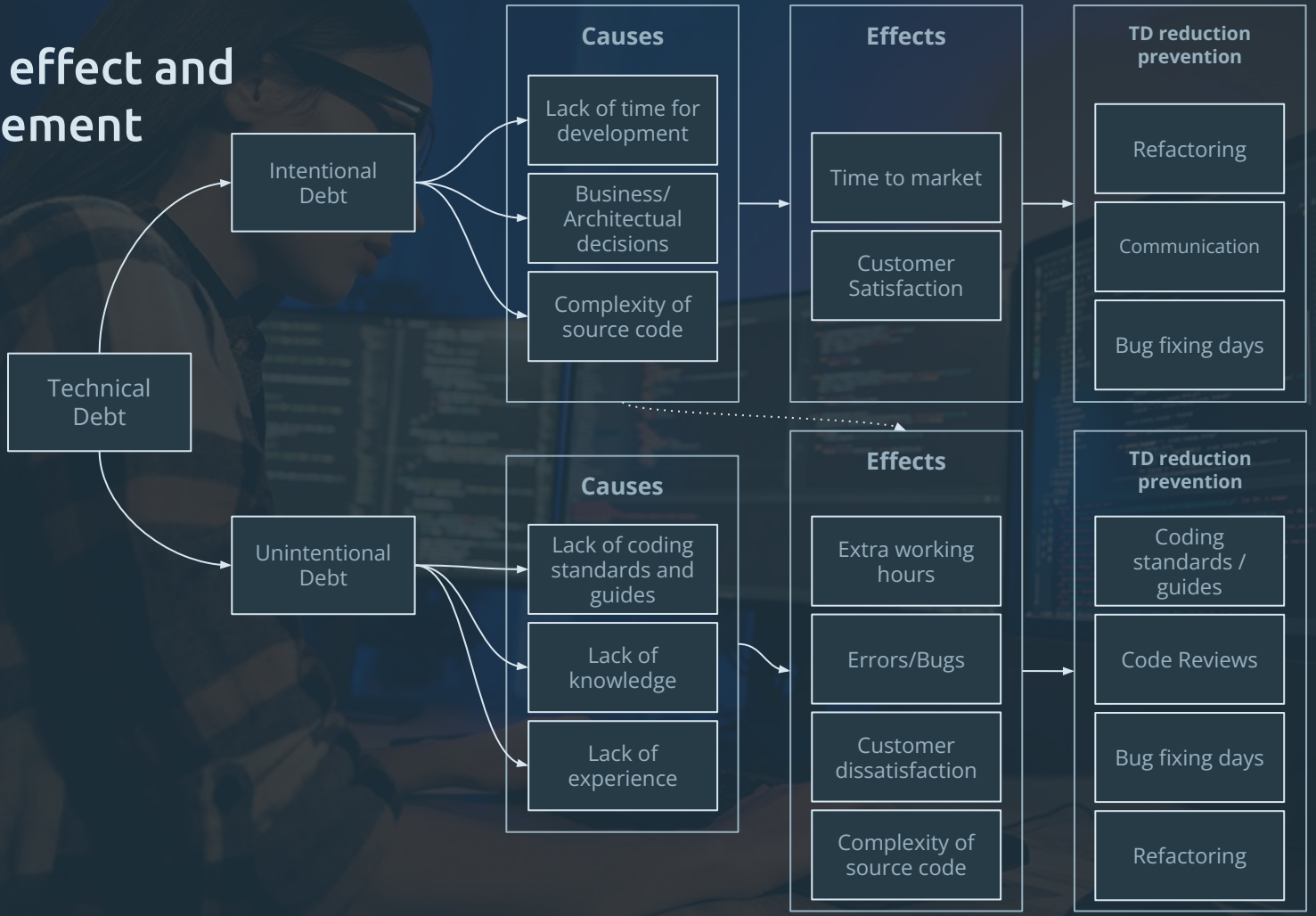
Product Roadmap

Internal Team Work

End Product

VISMA

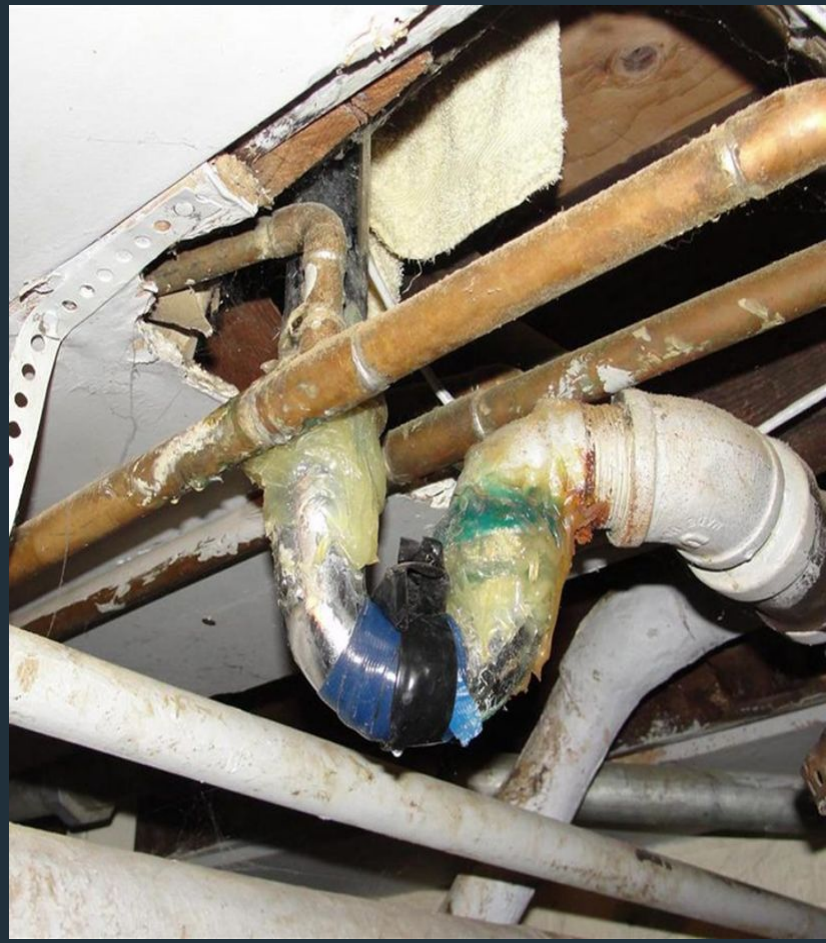
Cause, effect and management



VISMA

Why should anyone care?

VISMA



The Verge Menu +

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 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.

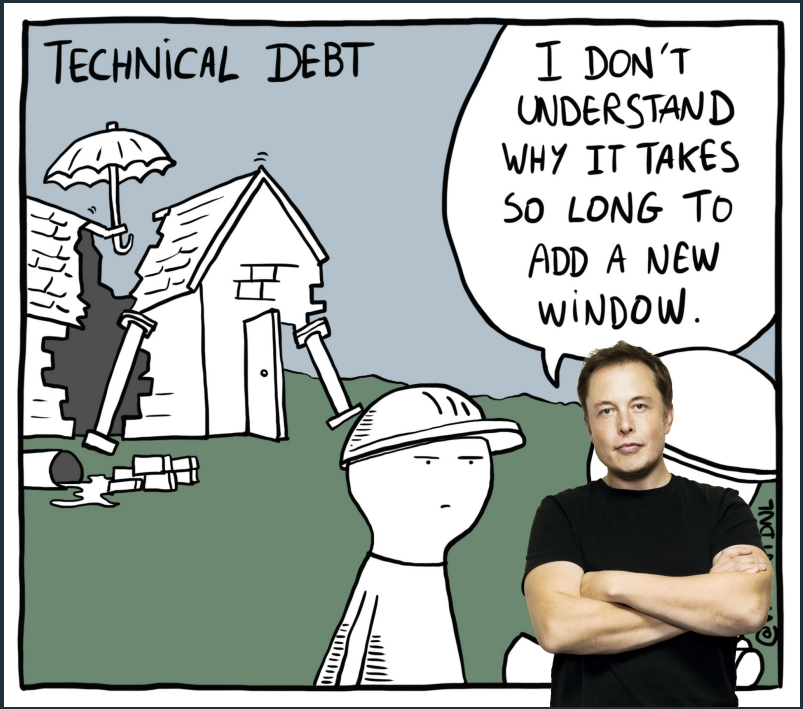
"There's so much tech debt from Twitter 1.0 that if you make a change right now, everything breaks," one current employee says.

The Verge 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.

VISMA


TECHNICAL DEBT
I DON'T UNDERSTAND WHY IT TAKES SO LONG TO ADD A NEW WINDOW.

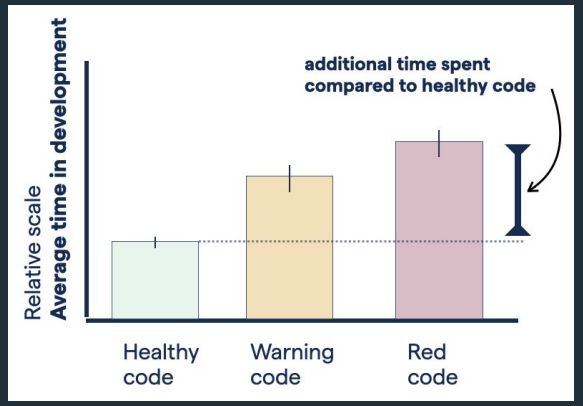
VISMA

Technical debt causes an **average productivity loss of 40%** (peaking at 90% in some projects)[1], **costing the world $5.82 trillion** (5 820 000 000 000) [2]

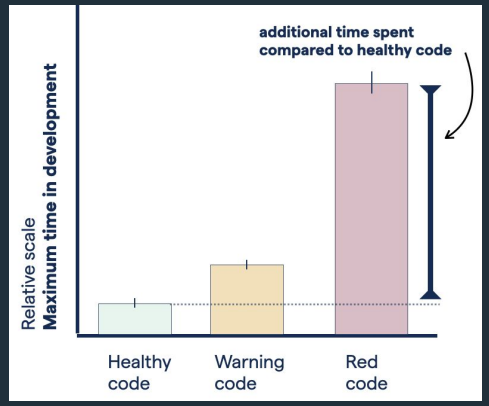
1 - T. Besker, A. Martini, and J. Bosch, 'Software developer productivity loss due to technical debt—A replication and extension study examining developers' development work', Journal of Systems and Software, vol. 156, pp. 41–61, Oct. 2019, doi: 10.1016/j.jss.2019.06.004.
2 - CISQ Consortium for Information & Software Quality I The Cost of Poor Software Quality in the US: A 2020 Report, page 42

VISMA

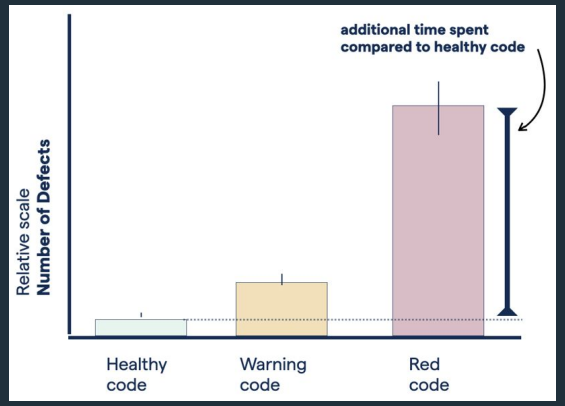
Recent studies from 39 commercial codebases



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

https://codescene.com/blog/measuring-the-business-impact-of-low-code-quality

Measuring Technical Debt

Static Code Analysis (SonarQube)

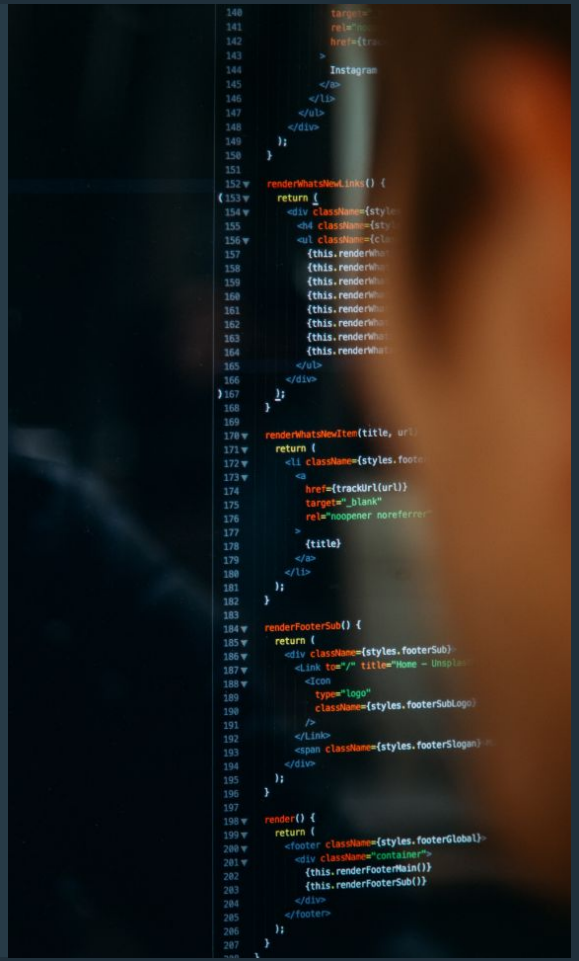
- **21.185 days or 58 years** of technical debt, for 1623 projects
- In **March 2023, 241 days** of effort of technical debt was introduced

Should not be the only way to measure technical debt

VISMA

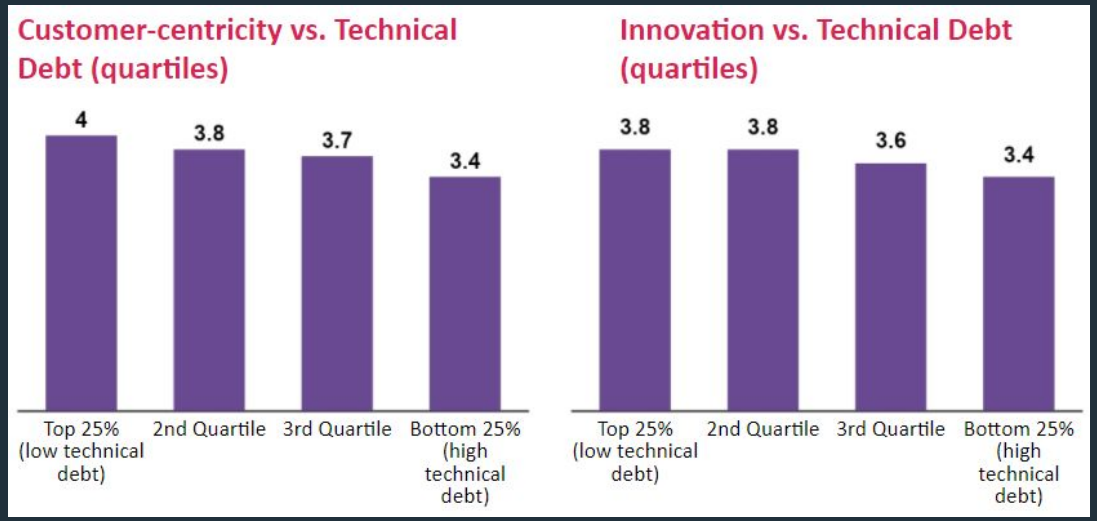
Why Technical Debt Matter?

Lessons learned from own research



VISMA

Technical debt matters!



Products with **more technical debt** are developed by teams with **lower customer-centricity and innovation** scores.

Presumably, it's difficult to have time and energy for being innovative and customer-centric when you have work with and around old technology and practices.

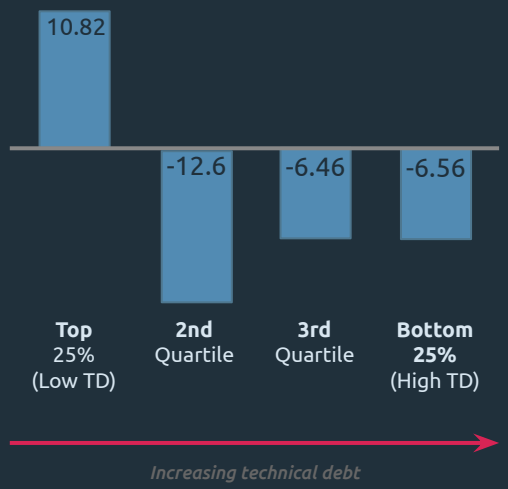
https://docs.google.com/presentation/d/1g6SKQexdPRZ0c5Q2k9A4typrexn8JJthAITjH7NLUmE/edit#slide=id.g14701d4b933_0_5

VISMA

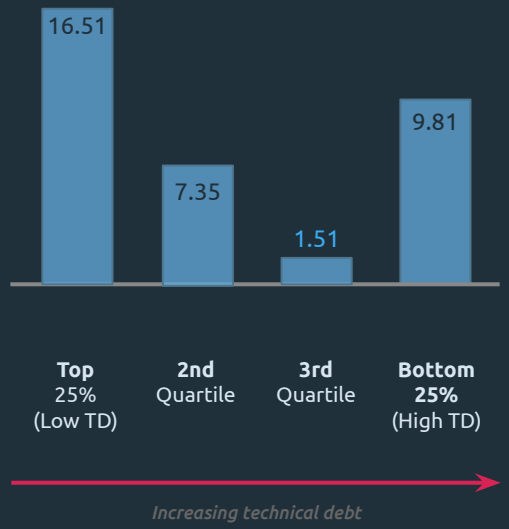
Technical debt matters!

The reduced innovation and customer-centricity in products with high technical debt drives down customer satisfaction and revenue growth.

1. Product NPS by technical debt (quartiles)



2. CBV Growth Rate (%) by technical debt (quartiles)



1. In general, products with less technical debt are more successful

- Higher product NPS
- Higher product growth rate

2. Product lines in the bottom 25% of technical debt have higher growth rate than expected. These product lines tend to have lower CBV, and could be smaller products that have accumulated technical debt through rapid growth.

VISMA

Creating awareness and working with Technical Debt

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

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

VISMA

Technical Debt Assessment

💱 Repayment

How is technical debt repaid?

👀 Prevention

How is introduction of unintentional technical debt prevented?

📐 Planning

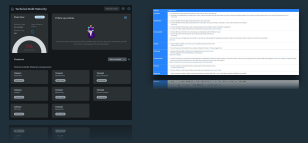
How is technical debt included and prioritized in the improvement plans?

✋ Identification

How, and where is technical debt identified?

📣 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?

VISMA

Example assessment

Workflow

Create

Create an assessment from the template

Do and learn

Go through the assessment, learn about the capabilities and do a gap analysis

Follow ups

Follow up on discovered improvements. Start to track technical debt issues

Review

Request a review with a reviewer

Complete

After async or in meeting review, assessment is set to completed

Re-assess

Continuously follow up through index, re-assessment will be recommended, if needed

VISMA

Technical Debt Maturity

Request review

Overview

In progress

Expiration date
Not specified

Repeats every
365

Open follow-up actions
0

0%
Overall progress

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

VISMA

Technical Debt Maturity

TDMA01 In progress

Prevention

Task 1 of 8

Prevention capabilites

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.

VISMA

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
100%

TDMA02
Identification
100%

TDMA03
Documentation
66%

TDMA04
Analysis
100%

TDMA05
Monitoring
Not started

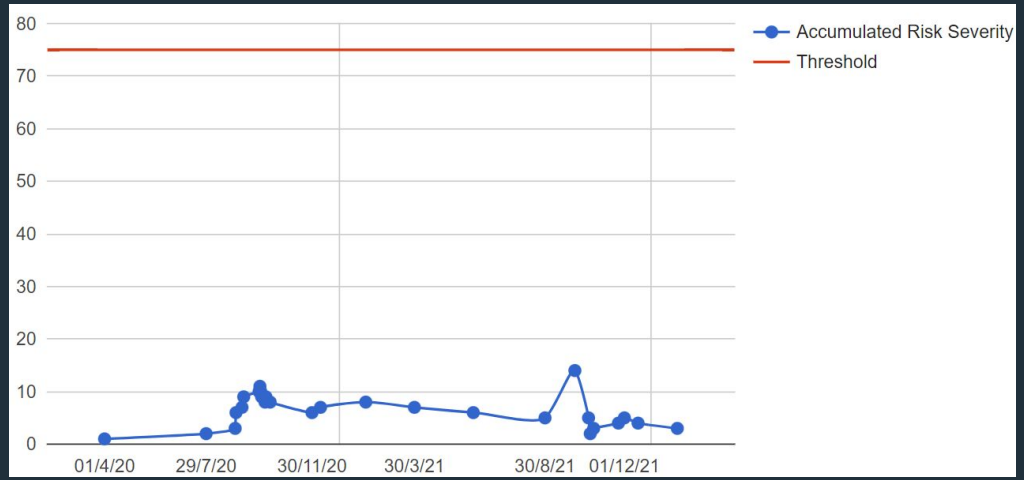
TDMA06
Communication
100%

TDMA07
Planning
50%

TDMA08
Repayment
100%

VISMA

Dashboard - team A



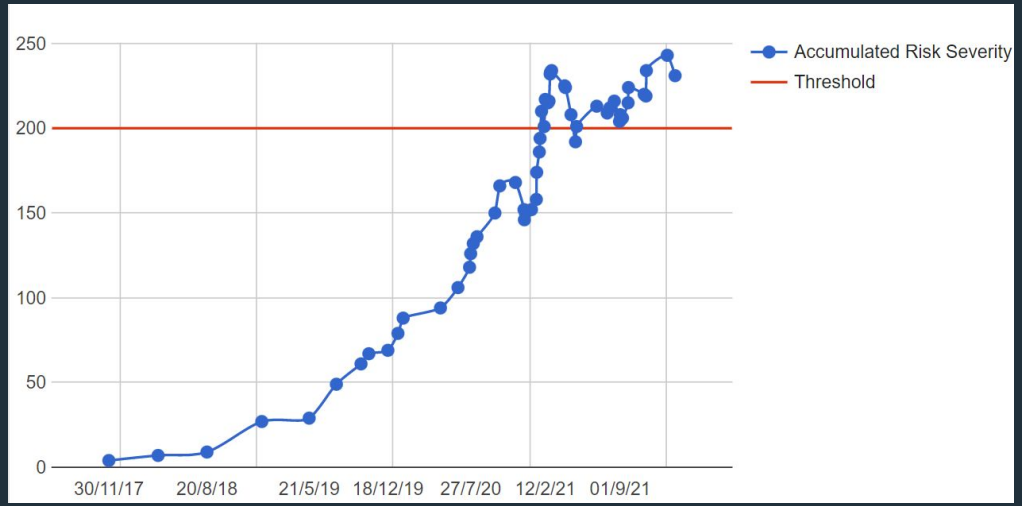
Filter Results: TDMA Example

T			P	Risk Impact	Risk Likelihood	Risk Severity ↓
			=	3	3	9
			=	3	1	3
			=	1	1	1
			=	1	1	1
			=	1	1	1
			=	1	1	1
			≪	1	1	1
			=	1	1	1
			=	1	1	1
			=	1	1	1
			=	1	1	1
			=	1	1	1

1–12 of 12

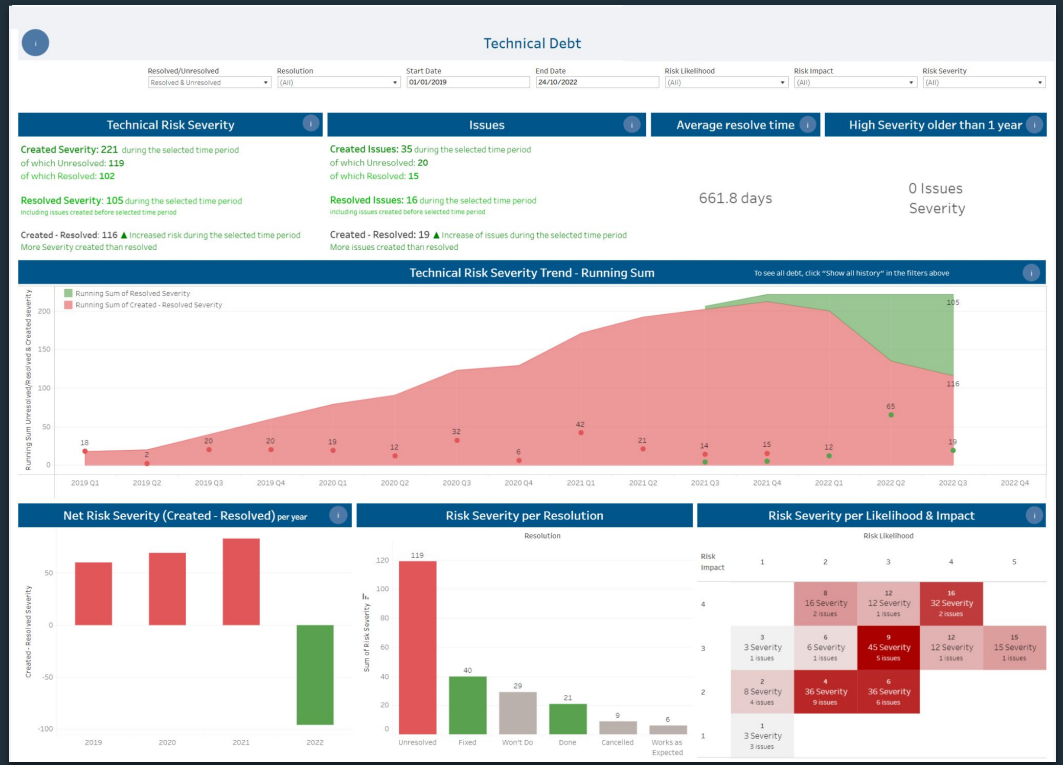
VISMA

Dashboard - team B



Filter Results: TDMA Example 2

T	Key	Summary	P	Risk Impact	Risk Likelihood	Risk Severity ↓
			=			
			=			
			^			
			=			
			=			
			=	4	4	16
			=	4	4	16
			=	3	5	15
			=	4	3	12
			^	3	4	12
			=	3	3	9
			=	3	3	9
			=	3	3	9
			^	3	3	9
			=	3	3	9
			=	3	3	9
			=	4	2	8
			=	4	2	8
			=	2	3	6
			=	3	2	6
			=	2	3	6
			=	2	3	6
			=	2	3	6
			=	2	3	6
			=	2	3	6
			=	2	2	4
			=	2	2	4
			=	2	2	4
			=	2	2	4
			=	2	2	4
			=	2	2	4
			=	2	2	4
			=	2	2	4
			=	2	2	4
			v	3	1	3
			v	3	1	3
			v	2	1	2



Technical debt **dashboard**

Out of the box

VISMA

Click on the sort icon beside each column header to sort the data. You can see, for example, Newest/Oldest issues by sorting on Created. Default sorting is on highest Risk Severity

Issues Details

Red Issues = Severity at least 15 and older than 1 year

Issue key	Project	Created	Resolution date	Days from Created to Today	Resolve time days	Risk Impact	Risk Likelihood	Risk Severity	Resolution	High Severity and older than 1 year	Status
		12/02/2021 11:28:51	14/04/2022 06:44:08		426	4	4	16	Done		Closed
		20/01/2021 07:58:54	28/06/2022 09:37:43		524	4	4	16	Fixed		Closed
		10/11/2021 10:29:39		348		3	5	15	Unresolved		Analysis
		30/06/2021 09:48:02		481		4	3	12	Unresolved		ON-HOLD
		07/10/2019 11:59:12	26/01/2022 08:20:15		842	3	4	12	Fixed		Closed
		23/09/2021 06:48:25	05/07/2022 13:24:20		285	3	3	9	Cancelled		Closed
		07/05/2021 08:09:37		535		3	3	9	Unresolved		Open
		28/01/2020 16:57:28	07/06/2022 11:47:57		861	3	3	9	Won't Do		Closed
		14/01/2019 14:34:16		1379		3	3	9	Unresolved		Open
		14/01/2019 14:32:57		1379		3	3	9	Unresolved		Open
		27/07/2020 11:06:04		819		4	2	8	Unresolved		Open
		14/01/2020 13:52:13		1014		4	2	8	Unresolved		Open
		30/09/2020 14:14:55	29/07/2022 10:19:23		667	3	2	6	Works as Expec...		Closed
		03/08/2020 11:24:51		812		2	3	6	Unresolved		Open
		23/06/2020 14:39:11		853		2	3	6	Unresolved		Selected for Development
		07/05/2020 12:46:25		900		2	3	6	Unresolved		Open
		28/10/2019 17:05:41		1092		2	3	6	Unresolved		Open

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 fai

On New Code

1 N

MEASURES

New Code

Overall Code

QUALITY GATE STATUS

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

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

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

VISMA

Examples from SonarQube

'System.Exception' should not be thrown by user code.
Code Smell

'System.Exception' should not be thrown by user code.
Code Smell

Refactor this code to not nest more than 3 control flow statements.
Code Smell

'System.Exception' should not be thrown by user code.
Code Smell

Remove the unused local variable 'request'.
Code Smell

Fix this implementation of 'IDisposable' to conform to the dispose pattern.
Code Smell +2

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

Remove the unused local variable 'request'. Why is this an issue? last year L73
Code Smell Major Open [redacted] 5min effort Comment unused

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

VISMA

“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

VISMA

“Afterall, won’t do”

Example of a technical debt

- External factors
- Time
- Changing priorities

“‘Integration’ is written in .NET 5 but support for .NET 5 ended on May 2022 after which no security updates are released for this version of .NET. Plan to upgrade to .NET 6.”

Conclusion: “Going to use another sync mechanism which will make this API obsolete, and we will delete it.”

.NET

VISMA

Do all teams in Visma work like this?

- Most of the teams working in a modern way (Visma Cloud Delivery Model) do this
- There is no one size fits all
- It depends

VISMA

Automatically track deviations

Visma Index

VISMA

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	

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

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

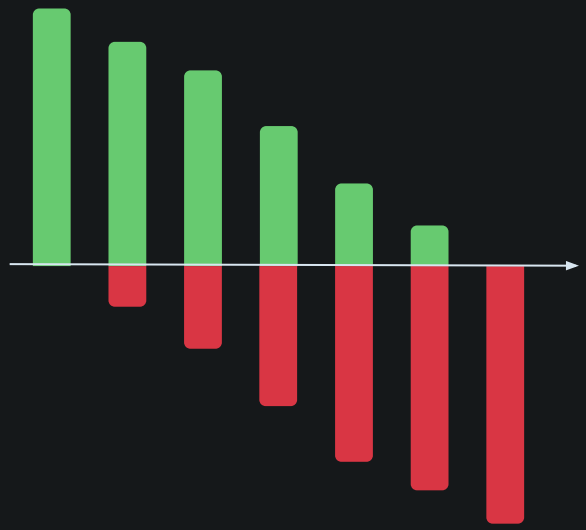
Research

- 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

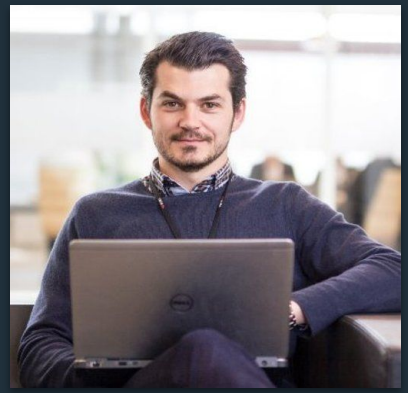
VISMA

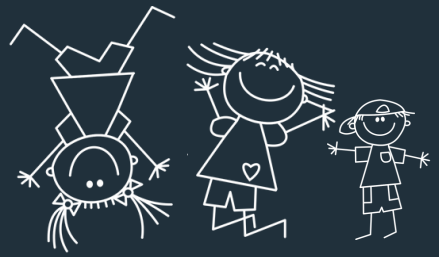
Summary

- **Visualise** your technical debt, so you know about it
- Think about **what makes sense** for your service (mature vs startup)
- Having **some technical debt is ok**, keep it under control
- **Never** try to **repay all** technical debt
- Focus on **actual debt first**, then potential debt
- Don't treat the **symptoms**, find the root cause



VISMA



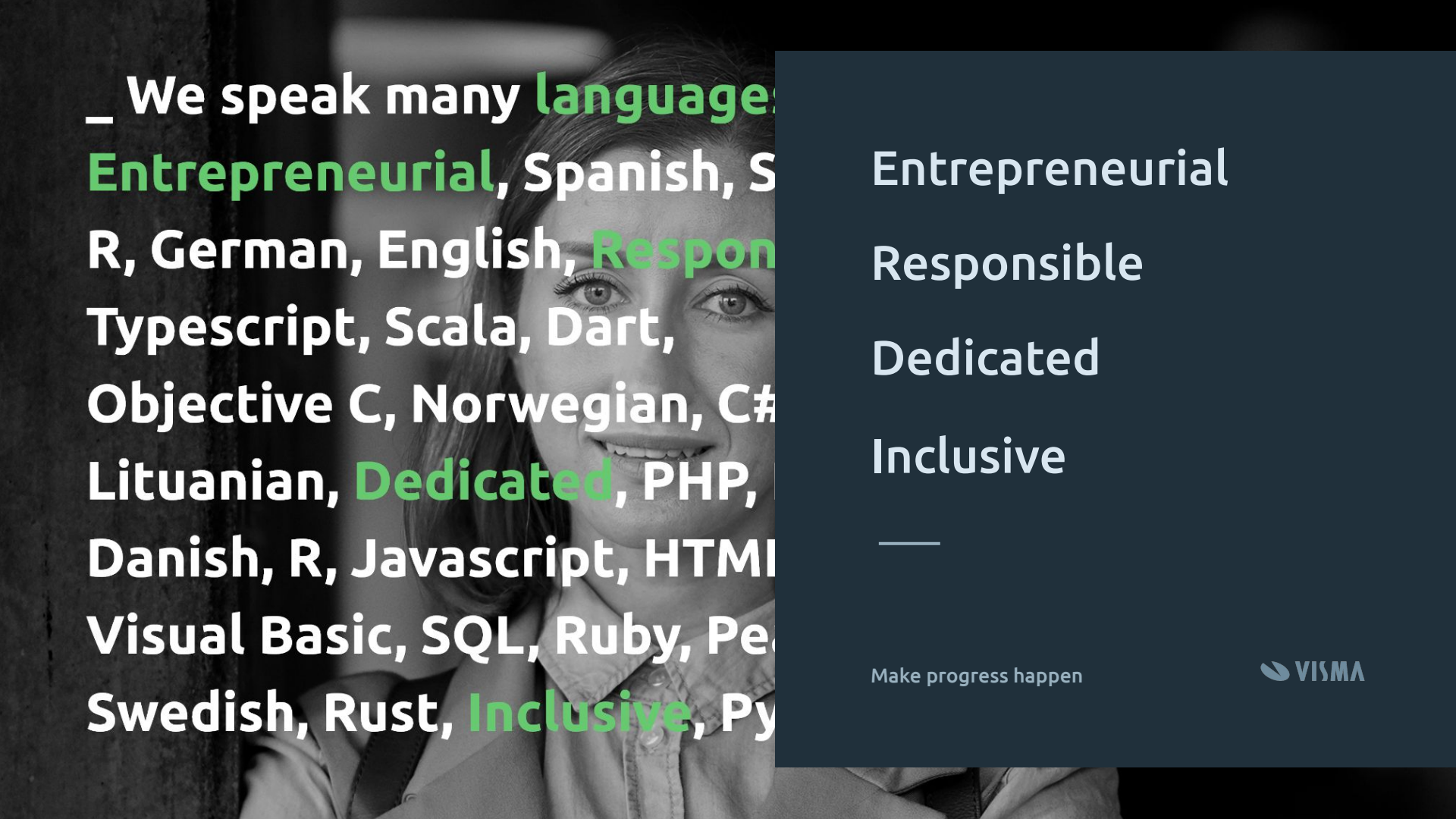


Mili Orucevic

Chief Software Quality Engineer

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

VISMA



Entrepreneurial

Responsible

Dedicated

Inclusive

Make progress happen

VISMA