Whitepaper

Code Red: The business impact of low code quality

This paper presents data from a large-scale study on how code quality impacts software companies in terms of time-to-market and product experience. We conclude with an analysis of the impact and specific recommendations towards successful software development.

Target audience

- Business managers
- Product owners/managers
- Technical managers
- Tech leads
- Development teams

About CodeScene

CodeScene is the intersection of code and people, empowering companies to build great software.

CodeScene was born in 2015 when founder Adam Tornhill published the book "Your Code as a Crime Scene". It introduced a new approach to software analysis which focused on the evolution of a codebase over time.

CodeScene has become the next generation of code analysis and is used by global Fortune 100 companies in a wide variety of domains.

CodeScene

Content

Key takeaways

Everyone in the software industry "knows" that code quality is important, yet we never had any data or numbers to prove it. Consequently, the importance of a healthy codebase is largely undervalued at the business level.
With this paper we remove the quotation marks so that "knows" becomes **knows** by attaching numbers on the impact of unhealthy code. That way, code quality can finally become the business concern that this study shows that it must be.

- Efficient software development is a competitive advantage that enables companies to maintain a short time-to-market with a mature product experience.
- However, research shows that 23-42% of developer's time is wasted due to Technical Debt and bad code.
- A key reason that this waste is tolerated is because code quality lacks visibility to non-tech stakeholders and possible gains in code quality are hard to translate into business value.
- This paper aims to elevate code quality to the business level by putting numbers on the impact of unhealthy code.
- Our research investigates 39 commercial codebases from various industries and domains. The finding are peer reviewed, statistically significant, and reproducible.[1] All metrics were automated via CodeScene.
- The results show that code quality has a dramatic impact on, both, time-to-market as well as the external quality of the product. High quality code has:
  A. 15 times fewer bugs,
  B. twice the development speed, and
  C. 9 times lower uncertainty in completion time.

Key findings for healthy code

✓ **15 times fewer defects**

✓ **Implement features twice as fast**

✓ **Reduce uncertainty in task completion times by an order of magnitude!**

[1] A. Tornhill & M. Borg (2022), "Code Red: The business impact of code quality - A Quantitative Study of 39 Proprietary Production Codebases". Accepted for publication in Proc. of International Conference on Technical Debt 2022

1. Introduction: a software tragedy

One of the great tragedies in software development is that code quality lacks visibility. Hence, it becomes far too easy to trade short-term wins like new features for the long-term maintainability of the code base.

This puts the business at risk: efficient software development is a competitive advantage that enables companies to maintain a short time-to-market without compromising the quality of their products. Adding to that challenge, our industry also faces a global shortage of software developers; demand substantially outweighs supply.

So at the same time that our industry is struggling with recruiting enough talent to meet ever shorter product cycles, research indicates that up to 42% of developers' time is wasted dealing with technical debt.

This implies that there is an untapped potential for software projects if the code quality is improved and technical debt paid down.

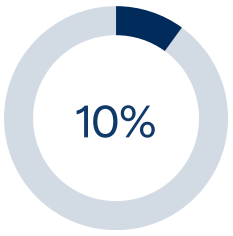
Until now, the business impact of code quality has been vague and frequently dismissed as a technical concern. Our mission is to change that view: in this paper we present our findings from studying 39 proprietary production codebases with respect to lead times for new features and the business risks in terms of defects.

Our results indicate that improving code quality could free existing capacity; with 15 times fewer bugs, twice the development speed, and 9 times lower uncertainty in completion time, the business advantage of code quality is unmistakably clear.

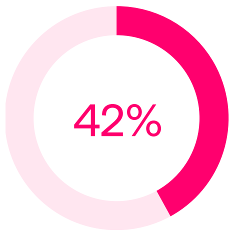
"With 15 times fewer bugs, twice the development speed, and 9 times lower uncertainty in completion time, the business advantage of code quality is unmistakably clear."

2. Where we are: the lack of code quality measurements

Without an industry-wide standard, code quality has remained a subjective and abstract concept that fails to gain traction at the business level.



Only 10% of business managers actively manage technical debt.



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

The costs of Technical Debt and poor code quality are well known.[2, 3] In the face of this trillion dollar problem, it's surprising that we still don't have any business level KPIs around technical debt. The lack of clear and quantifiable benefits makes it hard to build a business case for code quality and, hence, it's unfortunately much easier to trade short-term wins like new features for long-term sustainability.

Well-defined code quality KPIs would allow all stakeholders to assess the situation, estimate the business impact, and prioritise accordingly, similar to how financial KPIs drive businesses today.

We believe that there are two main reasons behind the absence of technical debt KPIs:

1. code quality is highly subjective, and

2. the industry lacks an established relationship between code quality and business outcomes.

Why is it so hard to measure the business costs of low code quality?

It's important to point out that the technical debt costs referenced so far are based on surveys and self-reported estimates and, to the best of our knowledge, no prior research has measured the relationship between development speed and code quality.

Part of the reason is because existing models for technical debt (e.g. SQALE and SIG TD) lack a measure of the actual business impact; the cost of technical debt is **not** the time it would take to fix the code — that's the remediation work -- but rather the additional development work due to technical quality issues.

To successfully crack this problem, we need to establish a link between a reliable code quality measure and the impact on the business.

[2] Besker, T., Martini, A., Bosch, J. (2019) "Software Developer Productivity Loss Due to Technical Debt"

[3] https://codescene.com/technical-debt/whitepaper/calculate-business-costs-of-technical-debt.pdf

3. Hard facts: data on the business impact of code quality

Everyone in the software industry "knows" that code quality is important, yet there is little data supporting that claim.

Without quantifiable values, we — as an industry — are unlikely to bridge the existing communication chasm between engineering and business. This is evident in the previously referenced studies: even though technical debt wastes up to 42% of developer's time, less than 10% of all business managers actually manage their technical debt.

But it gets worse. Much worse. A study by Besker et al. reports that "none of the interviewed companies had a clear strategy on how to track and address the wasted time"[4]. This implies that the majority of companies:

A. don't know how much time they waste on poor quality code and technical debt,

B. don't know where the main development bottlenecks are amongst millions of lines of code, or

C. don't have a strategy to reduce that waste.

The study and numbers presented in this white paper aim to change the situation by offering statistically significant data on the relationship between code quality and business outcome.

Development data from 39 companies

To measure the relationship between code quality and the business impact, we collected data from 39 proprietary codebases under active development. The collected data represents all development tasks that were completed over the past 6-12 months, depending on codebase.

We included codebases from industry segments as diverse as retailing, construction, infrastructure, brokerage, data analysis, and more to ensure a representative sample. We also capture data that generalises across implementation technologies. Hence, our dataset includes codebases implemented in 14 different programming languages (Python, C++, JavaScript, C#, Java, Go, etc.).

All companies gave their consent to participate in the study, and provided us with access to their source code repositories and Jira product data.

The measurements are performed through the CodeScene tool, which automates the code quality classification as well as measuring the cycle time in development and the ratio of software defects. Let's look at the metrics and results.

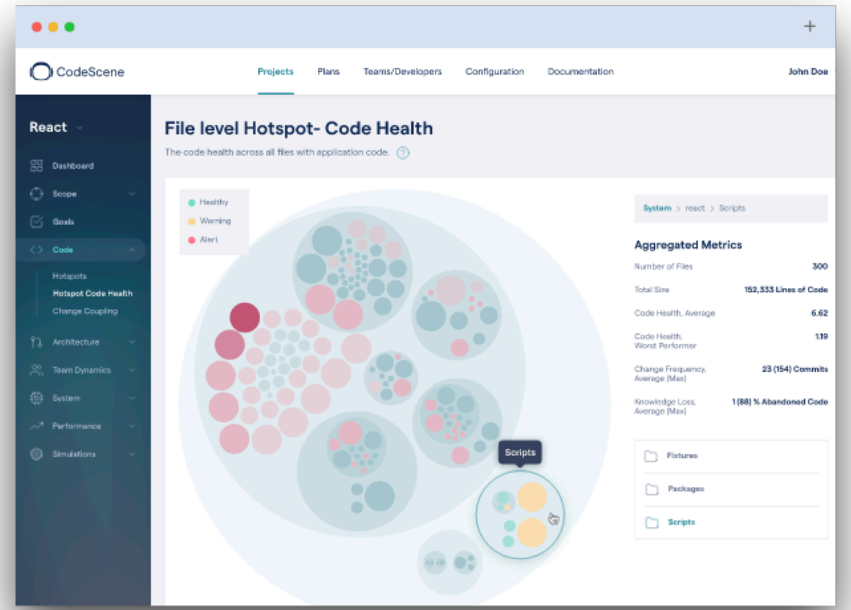
- 40,000 software modules
- 14 programming languages
- Multiple industry segments

[4] 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.

Measure Code Quality via the Code Health metric

Our research uses the Code Health metric as a proxy for code quality. Code health is an aggregated metric based on 25+ factors scanned from the source code. [5] The code health factors are known – from research – to correlate with increased maintenance costs and an increased risk for defects.
Based on the code health score, each source code file is automatically categorised by the CodeScene tool as:

- Green Code (healthy code with low risk for maintenance issues),
- Yellow Code (problematic code), and
- Red Code (unhealthy code with high maintenance risks).



File level Hotspot- Code Health

The code health across all files with application code.

Aggregated Metrics

Number of Files	300
Total Size	152,333 Lines of Code
Code Health, Average	6.62
Code Health, Worst Performer	1.19
Change Frequency, Average (Max)	23 (154) Commits
Knowledge Loss, Average (Max)	1 (88) % Abandoned Code

Visualising code health: each source code file is classified based on maintenance costs and risks.

Measure the business impact via Time-In-Development & Defects

The business impact of low code health is measured by integrating information from product life-cycle tools like Jira. The CodeScene tool fetches Jira issues, maps them to source code files, and calculates two data sets:

1. Number of defects per source code file, and
2. Time-In-Development per file and Jira ticket.

Time-In-Development is the cycle time a developer needs to implement the code associated with a Jira backlog item. The cycle time is automatically calculated by CodeScene, and is defined as the time between when an issue is moved to an "In Progress" state and when the last code related to that Jira issue is committed in version-control.

From a business perspective, the number of defects impacts the product experience: software with a high degree of defects delivers a negative product experience, which in turn impacts customer satisfaction and increases the risk for customer churn.

A longer Time-In-Development represents waste that negatively impacts the roadmap execution. Further, if work in low quality code leads to more unpredictable delivery times, then that will lead to lower confidence in commitments and strain the coordination within the organisation. Uncertainty is the enemy of any software delivery.

✓ **Code health automatically identifies problematic source code**

✓ **The Time-In-Development can be calculated automatically: no administrative overhead for measuring waste & opportunities.**

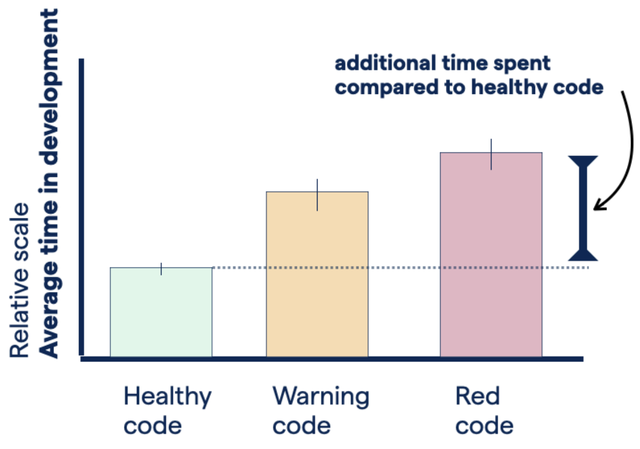
[5] https://codescene.com/code-health

124% faster development in Green Code

Our first research objective investigates the link between code health and time-to-market. We do that by measuring the average Time-In-Development for Jira tasks and correlating those numbers with the code health of the impacted source code files.

Our results show that implementing a Jira task in Green Code is 124% times faster than in Red Code for tasks of similar scope. This means that a feature that takes 2 weeks to implement could have been delivered in less than one week if the code had been healthy.

We also note a significant impact already at the Yellow code health level: the average development time for a Jira issue in Yellow code is 78% longer than in healthy code.



Implementing a feature or fixing a bug is twice as expensive in Red Code (relative scale).

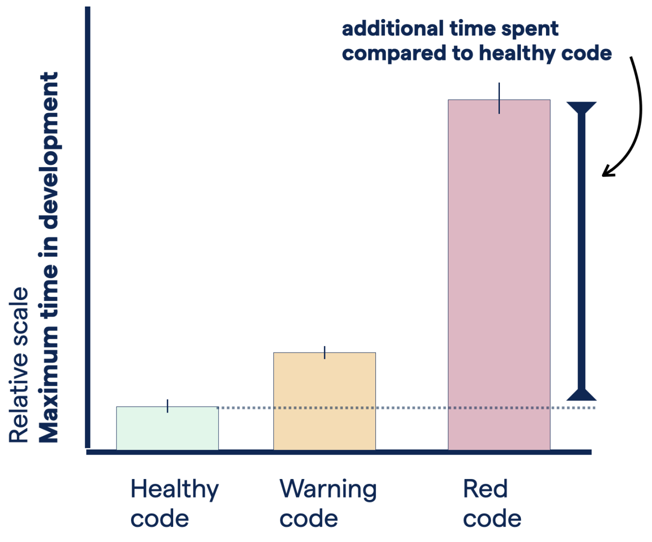
The uncertainty in Red Code means a feature can take an order of magnitude longer to implement

The additional development time for Red Code matches the intuition of experienced software developers: there's a productivity cost associated with low code quality. However, to us as a research team, the big surprise was that Red Code isn't just more expensive: it also seems to be more unpredictable than healthy code.

We explored that by measuring the maximum Time-In Development for each source code file. The results show that the maximum time to implement a Jira issue in Red Code is an order of magnitude larger than in healthy code!

Translated to a business context, an order of magnitude difference in completion time means very high uncertainty. As a product owner, high uncertainty makes it impossible to keep any commitments; be it to customers or internal stakeholders. And to a developer, uncertainty causes stress, over-time, and missed deadlines.

In essence, Red Code is not only inherently more expensive: it also carries a significant business risk.



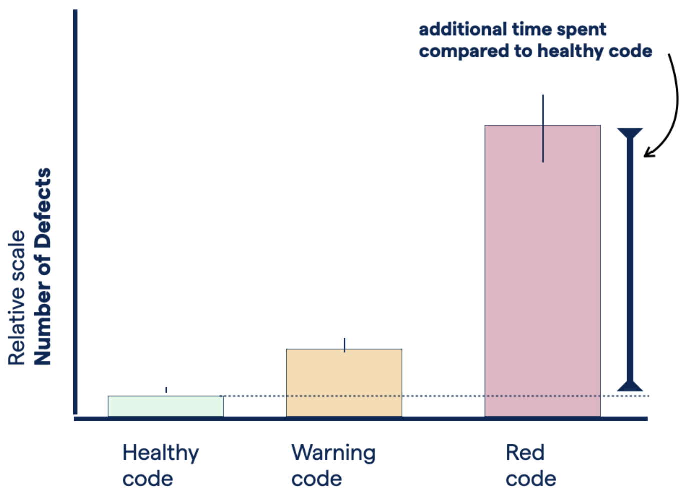
Red code: more than 9 times longer average maximum time leads to uncertainty during development (relative scale).

15 times more defects in Red Code

In addition to the excess development costs, we also wanted to explore the external quality perspective: does Red Code contain more bugs then code of higher quality?

The results are quite dramatic: Red Code has fifteen times more defects on average than healthy code! Just like for the maximum Time-In-Development, the relationship isn't linear: Red Code is significantly more problematic than Yellow code. That said, even Yellow code comes with a real cost and has, on average, 4 times as many defects as healthy Green Code.

This finding adds another business dimension, namely customer satisfaction. A high degree of defects also impacts the development team in the form of unplanned work. Bug prone code makes it hard to stay focused on planned tasks, which in turn causes additional waste via context switches.



Red code: 15 times more defects compared to high-quality code (relative scale).

All findings are statistically significant

All findings reported in this paper are statistically significant. That means, in layperson terms, that the findings are unlikely to be just a fluke or randomness in the data.

This is important: when advising on professional software development, money, jobs, and people are impacted. Hence, it's our responsibility as researchers to make sure our data and resulting recommendations are as accurate and reliable as possible; the software profession is still a relatively young field in need of more facts and fewer opinions.

The formal research publication (A. Tornhill & M. Borg, Proc. of the International Conference on Technical Debt, 2022) — the foundation for this white paper — defines the details of the data collection, analysis and statistical methods. We also made the data public so that the results can be replicated.

✓ **Peer reviewed & accepted for the International Conference on Technical Debt 2022**

✓ **Backed by academic research in collaboration with leading software experts**

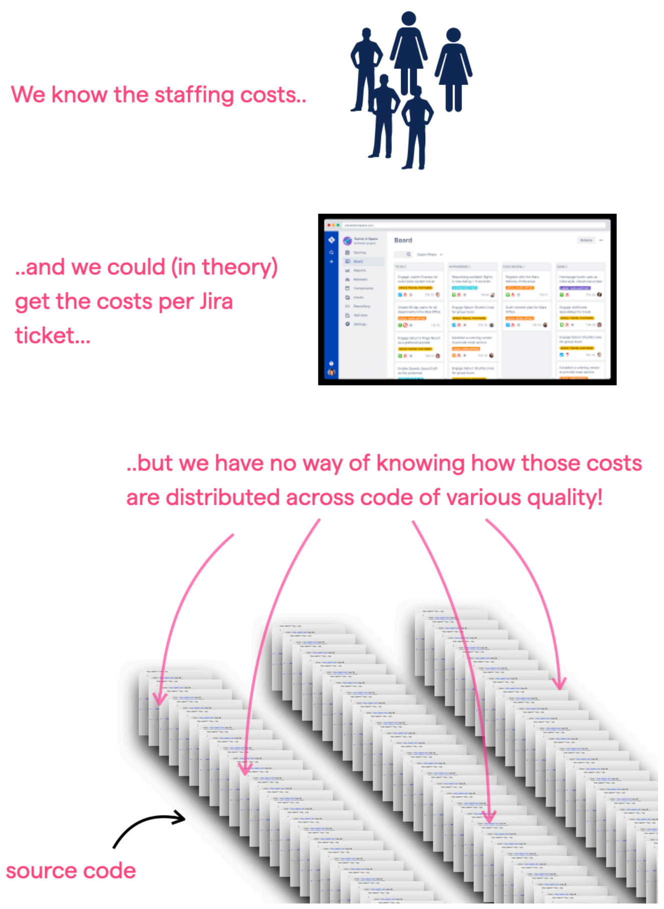
4. Impact: code quality enables successful software

Given that software companies waste up to 42% of developers' time on technical debt, it is surprising that as few as 7.2% of organizations methodically track technical debt[6].

Enable a data-driven approach to software development

One explanation for why this waste is tolerated could be simply that the impact of technical debt hasn't been possible to quantify at the level of the source code. Consequently, the few companies that do manage technical debt spend a significant amount of that time on identification and prioritization of potential issues to fix. Actual improvements are more rare, and the outcome uncertain.

The findings in this paper give us the option to challenge the status quo and elevate code quality to the level of a business KPI. More specifically, knowing — as a software company — where you have Red, Yellow, and Green Code enables a data-driven approach to software development.


We know the staffing costs..
..and we could (in theory) get the costs per Jira ticket...
..but we have no way of knowing how those costs are distributed across code of various quality!
source code

The efficiency loss due to low code quality hasn't been possible to assess at code level. The study in this report aims to change that by quantifying the costs of Red Code.

Increase developer productivity

Due to the predicted global shortage of software developers, companies will not be able to hire as many developers as might be needed; demand substantially outweighs supply.

However, with 15 times fewer bugs and twice the development speed, existing capacity is freed to fuel innovation and product growth. This is a clear and quantifiable business advantage that comes with healthy code.

[6] 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.

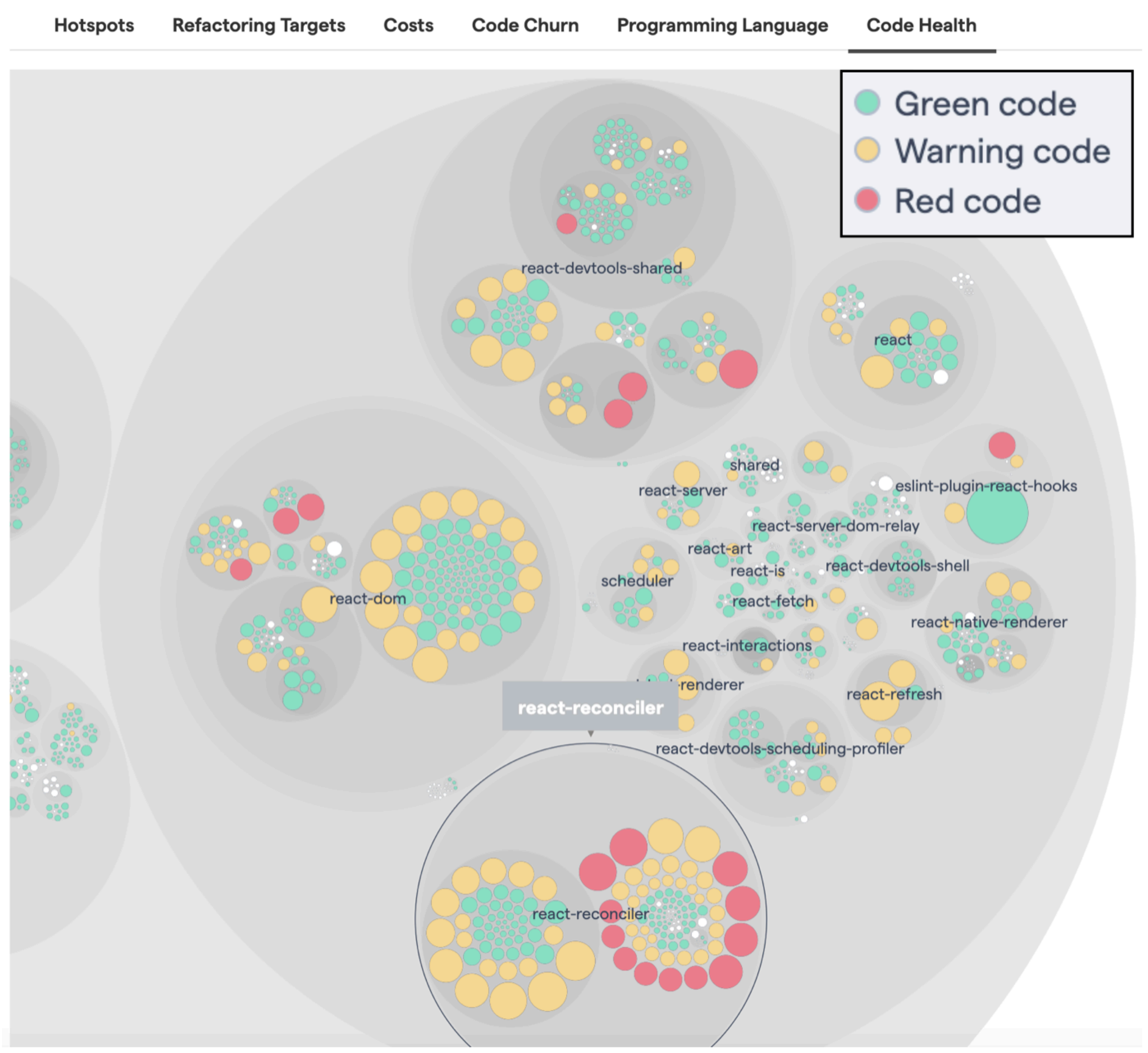
[7]Reduce uncertainty in estimates and commitments

Without visibility into your product's code quality it becomes too easy to ignore any warnings from engineering. Over time, the problems will deepen: features that took one day to develop a year ago now need 2 weeks, and often lead to unexpected rework. Low code quality kills innovation and progress.

Instead, knowing the health of your code reduces risk and aligns expectations. Consider a Product Manager (PM) responsible for the company's roadmap. If you — as a potential PM — knows the health of your code, then you'd use that when planning and prioritising features. Planning a feature that involves Red Code implies that the outcome is:

A. higher risk,
B. more expensive in terms of development time, and
C. a large uncertainty in completion time.

That way, you can make data-driven decisions on whether to go ahead with the feature as planned or, in case of code health issues, decide to start by refactoring/improving the existing code to reduce the risk.



Example of a code health visualisation of React.js from Facebook: we immediately see where the risks area (visualisation via the CodeScene tool).[7]

[7] https://codescene.com/blog/evaluate-code-quality-at-scale/

Speed + Quality — you can have it all

Software development productivity is a depressing topic: most proposed measures like velocity, added lines of code, commit counts, etc. have done more harm than good.

A prominent exception is the work by the DevOps Research Assessment (DORA)[9] that introduced productivity measures via its Four Key Metrics (FKM) . These metrics were also popularised through the Accelerate book[8], and are now spreading in the software industry.

The FKM of Accelerate clearly show that there is no trade-off between speed and quality. In fact, in order to deliver fast you also need high quality: the shorter your cycle times for deployment, the fewer production failures. This might sound counter-intuitive at first, but — as noted in this paper — our data suggests the same relationship for coding: the higher the quality, the quicker your development.

As such, one goal of our work is to complement DORA's delivery metrics with similar correlations between code quality and its business impact. As pointed out in the previous section, the waste during development can be significant.

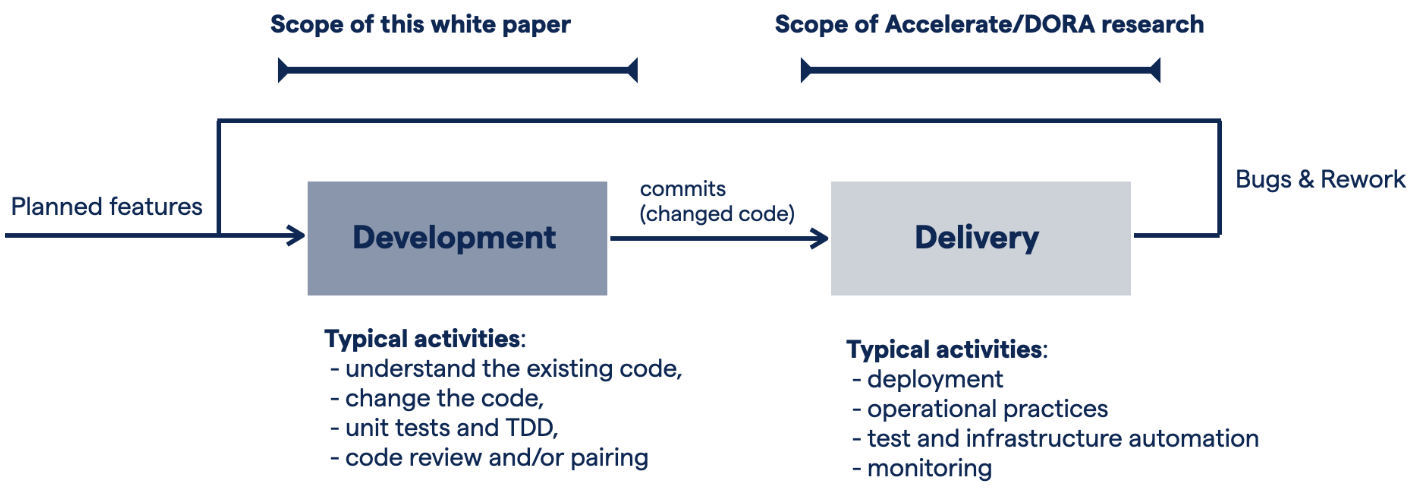


Figure. The scope of this white paper and how it complements the FMK from Accelerate.

"There is no trade-off between speed and quality. In fact, in order to deliver fast you also need high quality code."

[8] N. Forsgren PhD J. Humble and G. Kim. 2018. Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations. IT Revolution.

[9] https://www.devops-research.com/research.html

5. Summary: a call to put theory into practice

Key message

This research was initiated to make code quality a business concern by putting numbers on the impact of unhealthy code. It's an important topic since code quality has been an abstract concept that fails to get attention at the management level. Consequently, the software industry is wasting critical developer time on code that's more expensive to maintain than it should be.

Measuring code health enables data-driven decisions around core topics like Technical Debt, priorities, and roadmap risks. In particular, code quality improvements can come with a business expectation. That said, the impact of code quality goes well-beyond financial numbers; the inherent uncertainty in Red Code is likely to cause stress and friction within an organisation.

Finally, we acknowledge the fact that correlation doesn't imply causation. We plan to conduct future studies that help uncover the impact of code quality.

There's more:
Prioritise code health issues via Hotspots

Organisations need to balance short- and long-term goals. No matter how much we want, we simply cannot fix all Red and Yellow code at once. Instead, we need to prioritise by impact.

CodeScene's hotspots are a powerful tool for identifying the most expensive code quality issues. Read all about it here: https://codescene.com/blog/prioritize-technical-debt-by-impact/

Stakeholder benefits

The Code Health metric used in this research is automated and available via the CodeScene tool. That way, organisations can start measuring these KPIs today.

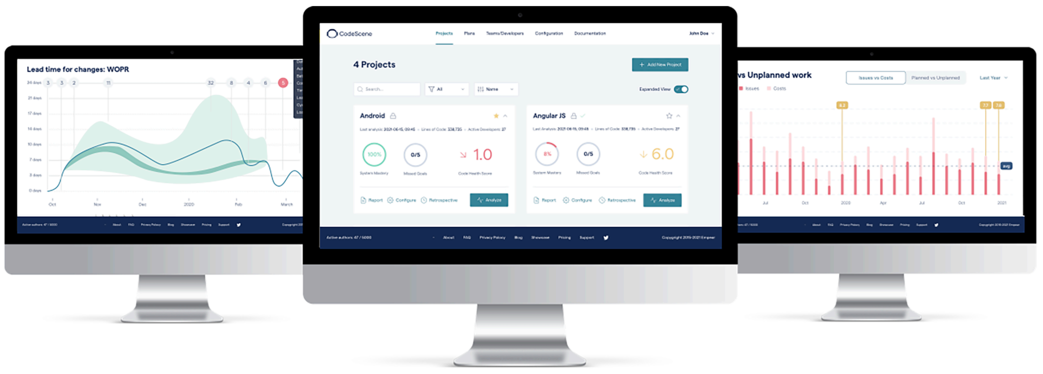
- **Business managers**: as evident from the findings in this paper, code quality constrains the business. Make Code Health a KPI that's tracked at the same level as pure business metrics like ARR, customer churn, EBIT, etc.

- **Product owners/managers**: manage risk and priorities via Code Health views. In particular, be conscious about the inherent uncertainty in Red Code. Use Code Health measures to balance the trade-off between new features vs improving what's already there.

- **Development teams**: these research findings offer a way to communicate improvements to the product and leadership teams. Many developers are forced to take on technical debt when the organisation pushes for wishful deadlines; use the Code Health views to communicate the risks using a terminology that means something to the business. Also, use Code Health trends[10] to show the negative impact influenced by accumulated technical debt.

[10] https://codescene.com/blog/3-code-health-kpis/

6. Further information



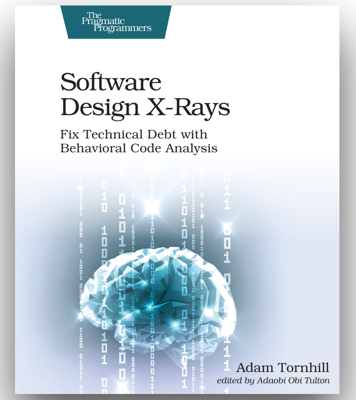
The CodeScene tool

CodeScene is a Swedish startup founded in 2015. CodeScene is the intersection of code and people, empowering companies to build great software. The CodeScene tool represents a new generation of code analysis and used by global Fortune 100 companies in a wide variety of domains.

CodeScene automates all the metrics covered in this paper.

About the author

Adam Tornhill is a programmer who combines degrees in engineering and psychology. He's the founder of CodeScene, and the author of Software Design X-Rays, the best-selling Your Code as a Crime Scene, Lisp for the Web and Patterns in C.


The Pragmatic Programmers
Software Design X-Rays
Fix Technical Debt with Behavioral Code Analysis
Adam Tornhill
edited by Adaobi Obi Tulton

Credits

A big, big THANKS to Markus Borg from Lund University for co-authoring the original research paper. Invaluable contributions!

Thanks to Aslam Khan, Joseph Fahey, and Romanela Polutak for reading drafts of this paper. Your feedback and comments made the paper so much better than what I could have done on my own.

This paper represents the progress towards a long-term goal of mine: make code quality a first class citizen of business. On this journey, I learned a lot from the technical debt community as well as from all the people that I had the chance to meet and discuss with over the years. Thanks!

Contact

Contact: adam.tornhill@codescene.com
Twitter: @codescene
LinkedIn: https://www.linkedin.com/company/codescene

Blog and articles

www.codescene.com
www.codescene.com/blog/

CodeScene