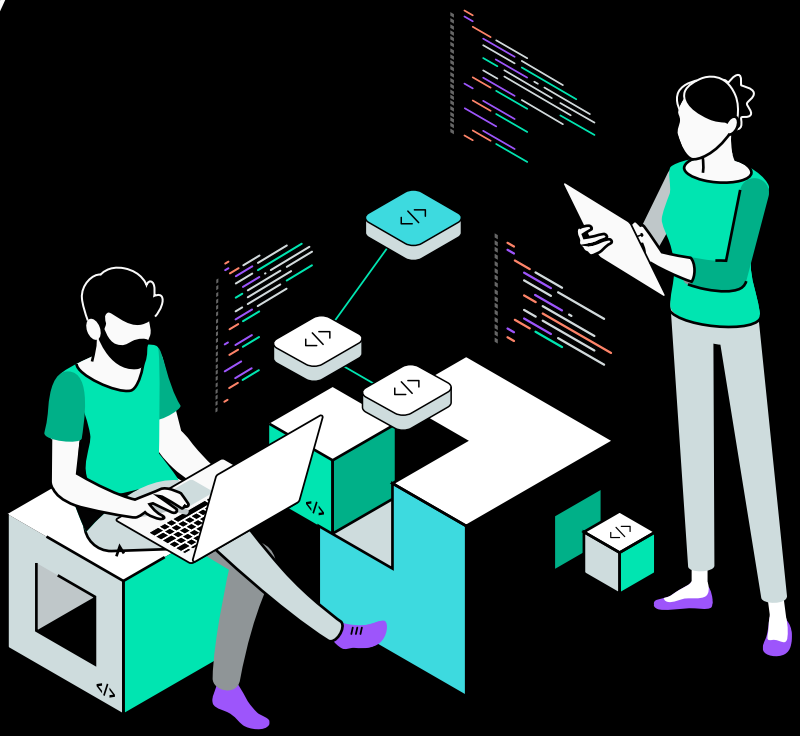
INCREDIBUILD

8 Ways to Enhance Developer Velocity

How to drive transformative business performance through developer velocity



Contents

Software ~~is Eating~~ has Eaten the World 3

What is Developer Velocity? 4

What's the DVI? 5

Why Developer Velocity? 5

What Influences Developer Velocity? 6

Eight Tips to Enhance Developer Velocity 7

Tip #1 – Learn to Fail Fast 8

Tip #2 - Build Product and Development Confidence 9

Tip #3 - Prototype 10

Tip #4 – Cathedral/Bazaar, Yes. Big Ball of Mud, No. 11

Tip #5 - Manage Technical Debt 12

Tip #6 – Manage Code Branches 13

Tip #7 - Maintain Traceability 14

Tip #8 – Beware Your KPIs 15

About Incredibuild 16

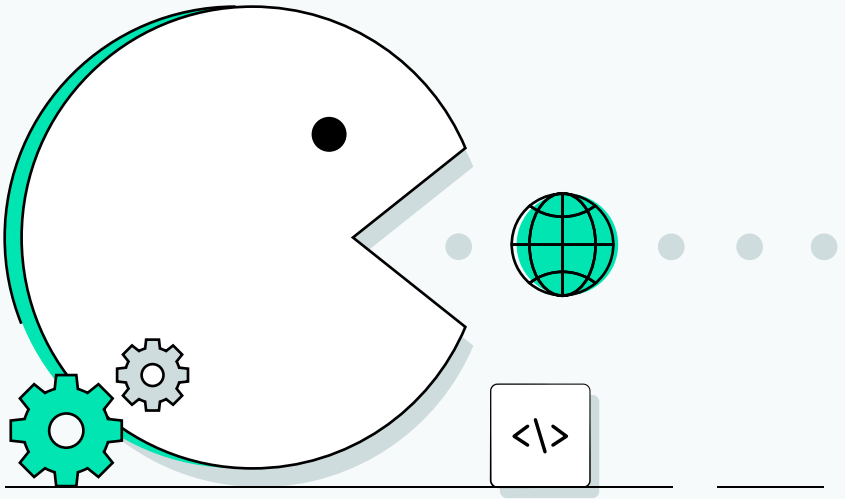
Software ~~is Eating~~ has Eaten the World

A decade ago, entrepreneur Marc Andreessen famously claimed that

> **“software is eating the world.”**

Today, that process is more or less complete.
Technology now powers enterprises of all shapes and sizes, touching all aspects of business and customer relations.

This means that today – in addition to their own core industry and domain competencies - nearly every organization needs some measure of software development capability.
Even organizations operating in distinctly non-tech industries like public transportation (think of Uber or any airline) or hospitality (think of AirBnB or any hotel chain) must develop a whole new range of software development skillsets in order to survive.



What is Developer Velocity?

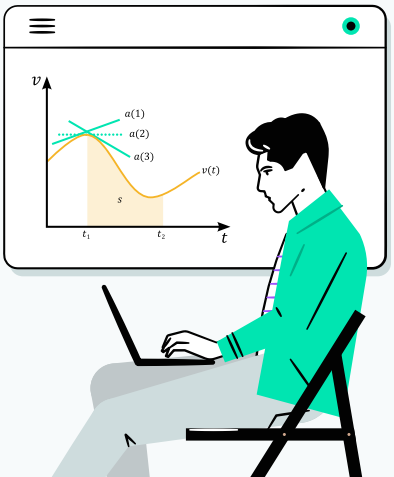
> Velocity (və-lŏs′ĭ-tē):
>
> A vector quantity whose magnitude is a body's speed and whose direction is the body's direction of motion, represented by the equation $v = \Delta s / \Delta t$

In physics, velocity is a well-known concept. In digital-centric businesses - where agility is mission-critical and keeping software current is key to agility - we talk about developer velocity.

Microsoft and McKinsey define developer velocity as

> **"the ability to drive transformative business performance through software development."**

To make this happen, organizations need to empower their in-house development team - crafting an environment that encourages innovation and reduces points of friction.



What's the DVI?

In early 2020, McKinsey surveyed nearly 450 large enterprises, quantifying the impact of Developer Velocity on business metrics and discovering which drivers had the most impact. The resulting index - the Developer Velocity Index (DVI) - expresses how well an organization is equipped to move digital ideas into production. The index also helps identify the most critical factors in achieving Developer Velocity.

Why Developer Velocity?

Microsoft found that across all industries, companies with high DVI:

X4-5
faster revenue growth

60%
higher shareholder returns

20%
higher operating margins

55%
higher score on innovation scales

Based on some 50 parameters - from architecture, to engineering practices, security and development tools, and more – McKinsey, too found a strong correlation between DVI and key business performance indicators.

THE BOTTOM LINE:

Companies with higher DVI are more agile, more responsive to market changes and customer demands, and ultimately better positioned for profitability.

What Influences Developer Velocity?

In order to deliver results, any skillset needs to the have the right combination of factors backing it up. McKinsey found that companies looking to master Developer Velocity need to empower developers, anticipate critical enablers, keep their investments aligned with customer value, and eliminate productivity barriers.

Github created its own Developer Velocity model, nicknamed SPACE, and concurs that the influences on developer productivity are varied and complex. To cut through the fog surrounding the issue, McKinsey analyzed nearly 50 individual performance drivers to determine what specific conditions created high Developer Velocity. They found that tools, culture, product management, and talent management had the greatest impact on business performance.

8 Tips to Enhance Developer Velocity

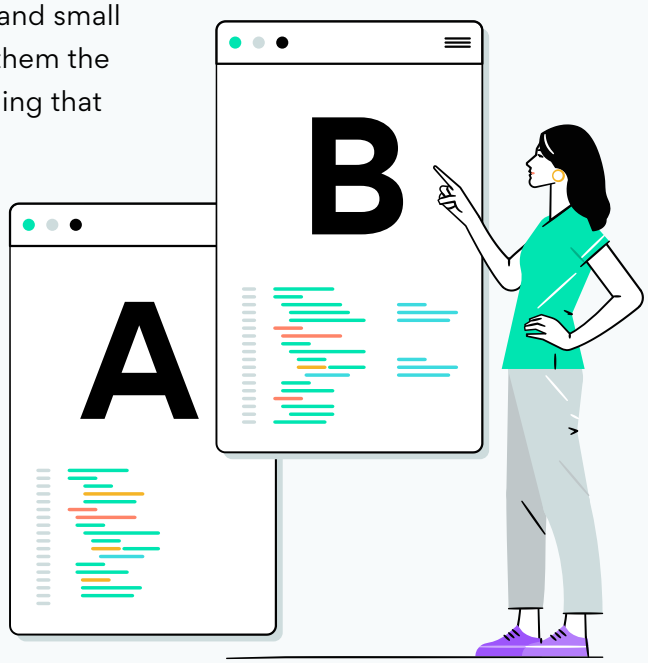
TIP

#1

Learn to Fail Fast

Intelligent organizations learn that failing is not a negative thing. Products fail when they don't correctly meet requirements or customer needs.
Why is this good? Because the faster products fail, the faster they can be corrected, and the more suitable they become to their target audience.

On the product level, AB Testing enables fast feedback on proposed changes in the product.
On the development level, Test-Driven Development (TTD), Static Code Analysis, and a Shift-Left approach empower developers to find problems earlier, fix them earlier and move much faster.
Tech and methodologies that encourage agility and small iterations offer developers a safety net – giving them the confidence to raise velocity with the understanding that course correction is built into the process.



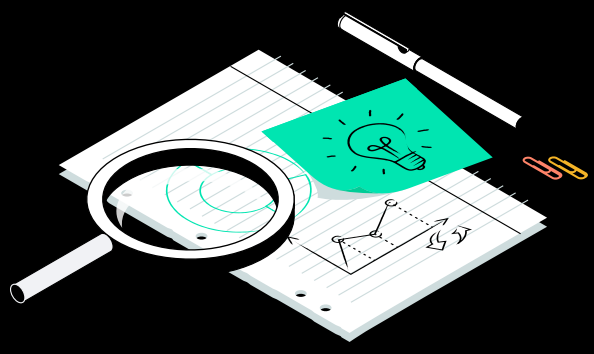
TIP

#2

Build Product and Development Confidence

Learning to fail fast as an organization is contingent on optimal product and development team confidence.
The reason? Like anyone in almost any situation, developers move more slowly when they do not feel confident.
By creating a culture in which mistakes are not just tolerated but encouraged, both product leads and developers gain the confidence to try new ways of doing things – many of which may be ultimately better and faster.

From a management point of view, raise confidence through error tolerance, encouraging a trial-and-error approach (while controlling investment in errors by failing fast), and remaining open to new ideas and unorthodox approaches. At the development level, consider adopting a pair programming methodology, and being liberal when you can (and strict when necessary) in code and design reviews.



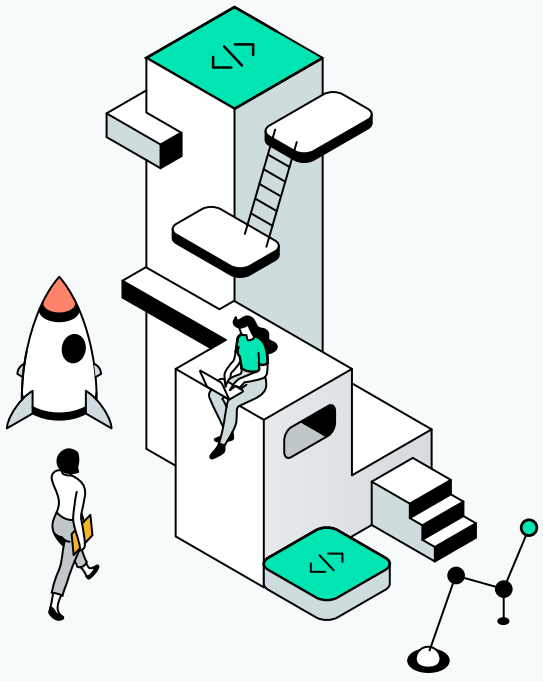
TIP

#3

Prototype

Although there is a school of thought that considers prototyping a waste of resources, the fact remains that it is less costly to create and discard a failed prototype than to do so with an actual product.

Prototyping reduces risks and builds knowledge – enabling developers to seek out and tackle risks safely in a small sandbox, rather in the actual code base.
This enhances developer velocity by helping developers get faster results and answers – even if these answers are no-go's.
Working on a prototype encourages the out-of-the-box thinking that leads to breakthrough innovation – and all without introducing noise into the actual product.



TIP

#4

Cathedral/Bazaar, Yes. Big Ball of Mud, No.

Conceptual integrity can be maintained whether development leaders choose a Cathedral or Bazaar development strategy – but nothing impedes developer velocity like a Big Ball of Mud (messy code and architecture).

Developers need clear architecture, good processes, effective ongoing internal communications, well-documented release notes, and an environment that discourages sloppy work. This encourages developer velocity by helping developers readily find the places in code with which new features interact, locate the root cause of bugs, understand the implications of each new part of the code, plan and understand better what and how to test.

TIP

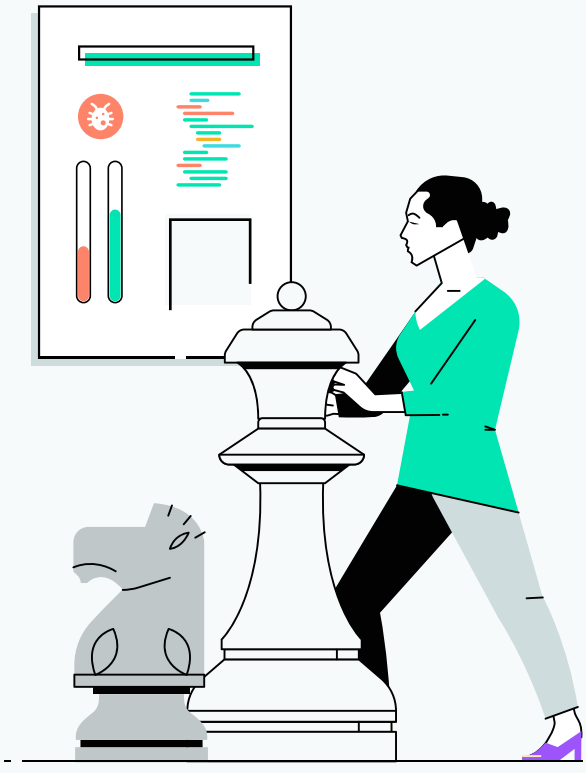
#5

Manage Technical Debt

No one plans to build a Big Ball of Mud. Technical debt is what happens when years go by developing on an old infrastructure, or after implementing many small changes in a hurry or without a proper plan, tools or libraries. It is a constant threat, and it is the enemy of developer velocity.

Technical debt means that developers have to invest significantly more effort to address each new feature or bug fix – lowering their motivation and raising frustration. Products with significant technical debt drive away versatile and quick people - making it much more difficult to leverage new technologies that speed development, lower time to market, and ultimately result in higher-quality products.

Whether your organization chooses to deal with technical debt by rebuilding the product from scratch while maintaining the old product, or just carefully rebuilding parts of the product inside the old code base - closing technical debt is imperative to achieving better developer velocity.



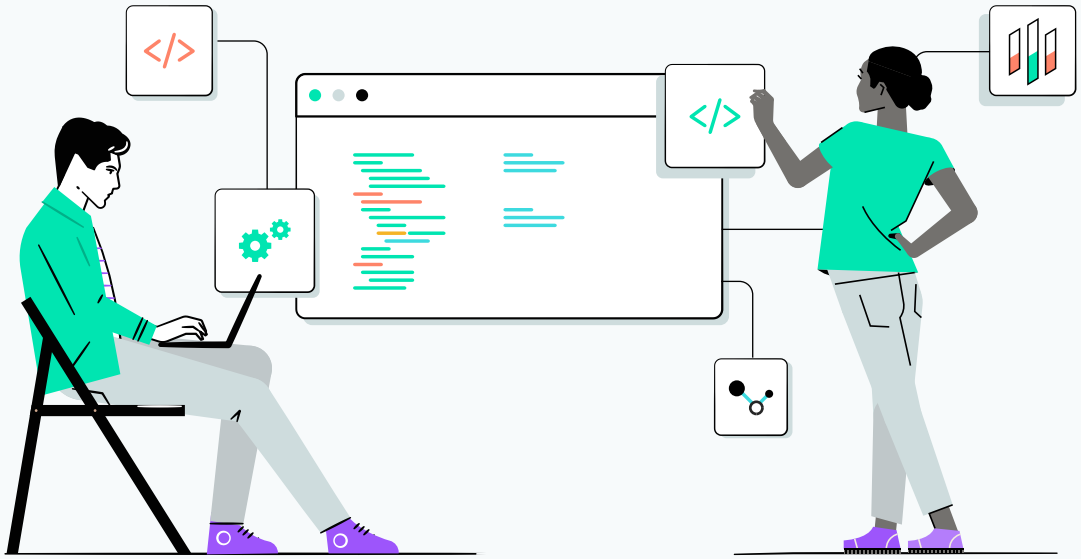
TIP

#6

Manage Code Branches

Branching is what allows software development teams to work in parallel.
Yet when developers need to spend time and effort chasing after the main branch owing to inflation of open branches or long living branches waiting to be merged into the main branch - velocity is in danger.

Effective branch management facilitates developer velocity by lowering the number of open branches dev teams have to maintain – and the amount of effort and (frankly) developer annoyance associated with this maintenance.
To avoid “chasing the master”, break big changes into small pieces whenever possible. For big chunks that seem unbreakable, finding the right refactoring approach (like adding a new level of abstraction) can enable breaking changes into bite-sized chunks that don’t affect code but do enhance developer velocity.



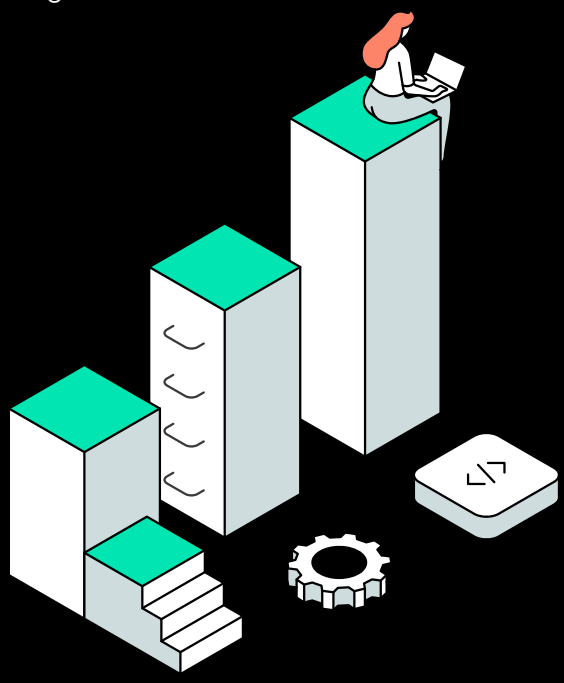
TIP

#7

Maintain Traceability

Traceability is a non-functional requirement, but one that should never be overlooked. Traceability encompasses effective logging and cross-system session management, and helps expose all relevant data in production environments in an easy-to-use way – dramatically impacting velocity.

Good traceability builds a much deeper understanding of the system, empowering developers to act faster and with greater confidence. While traceability does not, of course, replace testing and other required gates - developers confident in their ability to trace problems when they arise are less hesitant to quickly release new features and bug fixes.



TIP

#8

Beware Your KPIs

Product and development KPIs were conceived to allow management to follow development progress. Yet KPIs can also be false indicators of progress. Over-reliance on measurements can inadvertently incentivize developers to work toward the measurements themselves – making the metrics the target rather than a tool for reaching the target.

For example, code coverage in testing is important, but as a sole KPI it may lead to naïve tests that cover the code, but not actual scenarios of interest. Setting goals closer to actual targets allow your teams to get to these targets faster, as opposed to defining KPIs which may divert them down side paths.

About Incredibuild

Incredibuild created the industry's first hybrid acceleration platform for development processes - compilations, CI/CD builds, testing and more. The company's powerful distributed processing and unique build caching acceleration solutions turbocharge both dev cycles and iteration frequency - This enhances product quality, lowers time-to-market and raises customer satisfaction – all while dramatically lowering computing costs on-prem and in the cloud.

The only commercial tool bundled into Visual Studio. Incredibuild boasts over a quarter million users from some 2500 global organizations, including twenty Fortune 100 companies. For more information about Incredibuild, please visit www.incredibuild.com.