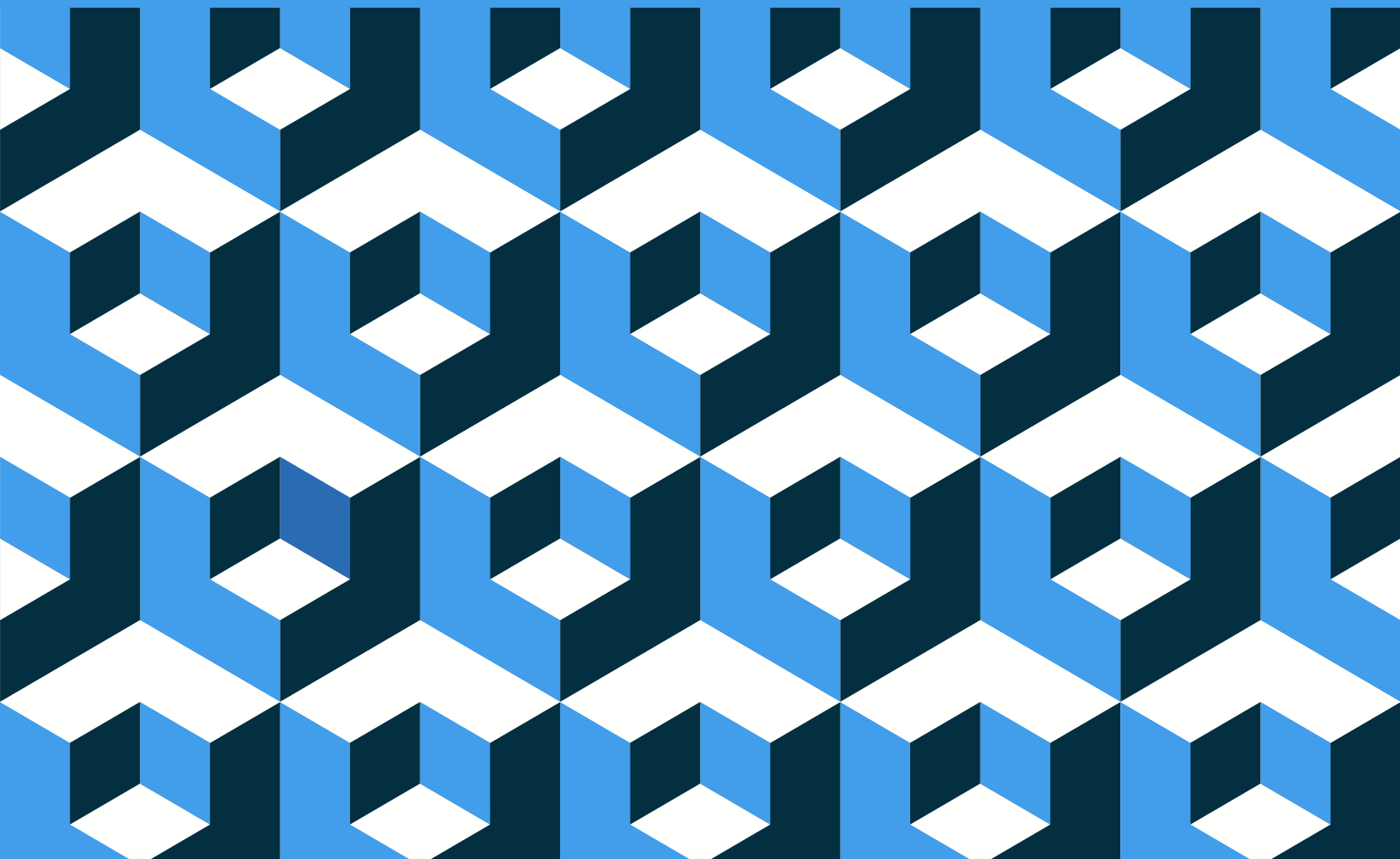
PLURALSIGHT

SOFTWARE ENGINEERING: WHY HAS IT ELUDED DATA-DRIVEN MANAGEMENT?

How engineering has been operating in the dark and what to do about it



OVERVIEW

Imagine yourself as a fly on the wall of an executive boardroom, listening to the heads of each corporate division present an overview of the past year:

Marketing will share a range of metrics on lead conversion, cost per lead and ROI on marketing spend; Sales will walk through a detailed, quantitative breakdown of the sales funnel; Finance will present a broad set of Key Performance Indicators; the operations and customer retention teams will similarly present a variety of critical metrics.

But, what about the software engineering organization? We can speak to features delivered, story points completed and ticket velocity… but these are all subjective measurements, not meaningful metrics based on hard data.

Even though much of a company's value is directly tied to their investment in software, most executives operate in the dark when it comes to understanding the performance of their engineering team. Instinct and 'gut feelings' are the best tools we've had to make decisions about budget items costing millions of dollars.

Just as other disciplines have benefitted immensely from objective metrics and actionable KPIs, forward-looking software organizations are experiencing the same lift that has transformed the rest of the enterprise.

Today, gut feelings are being replaced with data.

HOW WE GOT HERE

Software engineering teams are made up of analytical, intelligent individuals—so why aren't they as metrics-driven as other departments?

If you ask software engineering leaders why this is the case, you'll get a range of answers:

- Engineering is an art form—metrics cannot properly reflect productivity
- Engineering data is not easily accessible
- We measure story points and ticket velocity

Of the answers above, the most legitimate is that engineering data is not easily accessible. Until recently, data housed in git-based repositories has been challenging to leverage, particularly if an organization uses many repositories and if engineers use multiple aliases to commit code.

However, this issue has been solved by a recent class of applications that can measure and quantify data across git repositories.

With this hurdle cleared, the next question is how soon the software engineering world will embrace data-driven management, the way that every other organization in the executive boardroom has.

TURNING THE LIGHTS ON WITH DATA

Which brings us to the question of, if the software engineering world is truly next in line to embrace a metrics-driven management approach, what metrics will emerge as the industry standards?

The following is our take on the engineering metrics of the future, based on our work with over 300 customers:

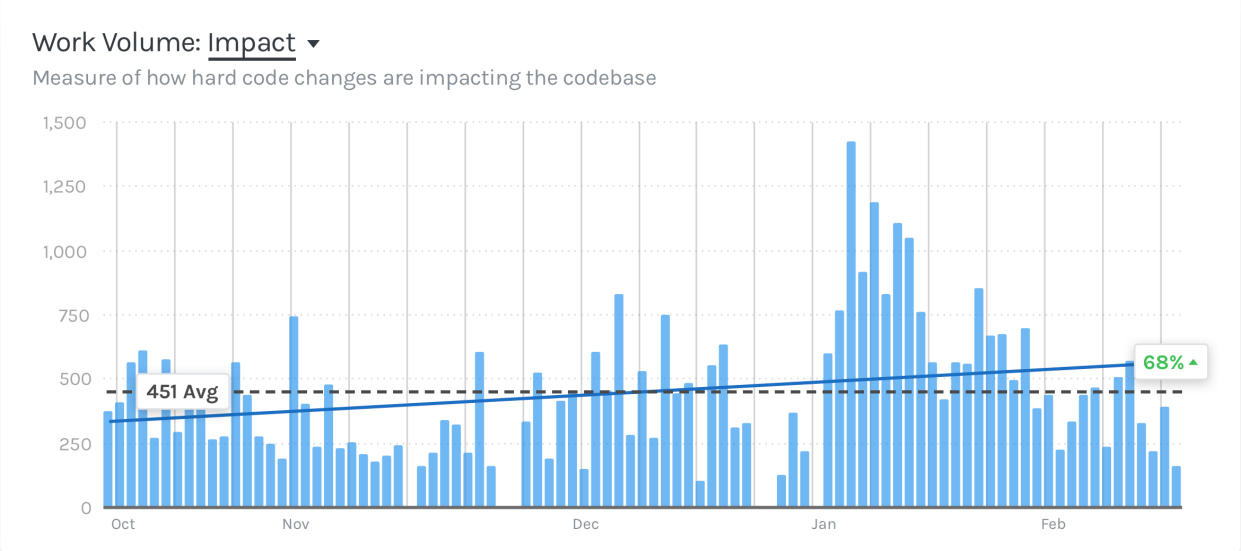
INSIGHT 01

Productivity and Output

The first question for a software engineering organization is whether its total output and productivity are improving, compared to prior periods. Software engineers are difficult to hire and often require ramp-up time, so increasing the output of an existing team is the most immediate way to move the needle on value delivered.

The most basic measurements of output, which are code volume (e.g. lines of code) and commit volume, are deficient and not complete indicators of the complexity or sophistication of work completed.

Therefore, we expect the industry to standardize around metrics that better reflect the cognitive load of work completed, such as Pluralsight Flow's Impact metric. Impact attempts to answer the question: "Roughly how much cognitive load was carried when implementing these changes?"*



Baselining past team output levels and measuring progress towards systematic improvement is a clean way for engineering teams to generate and document productivity gains.

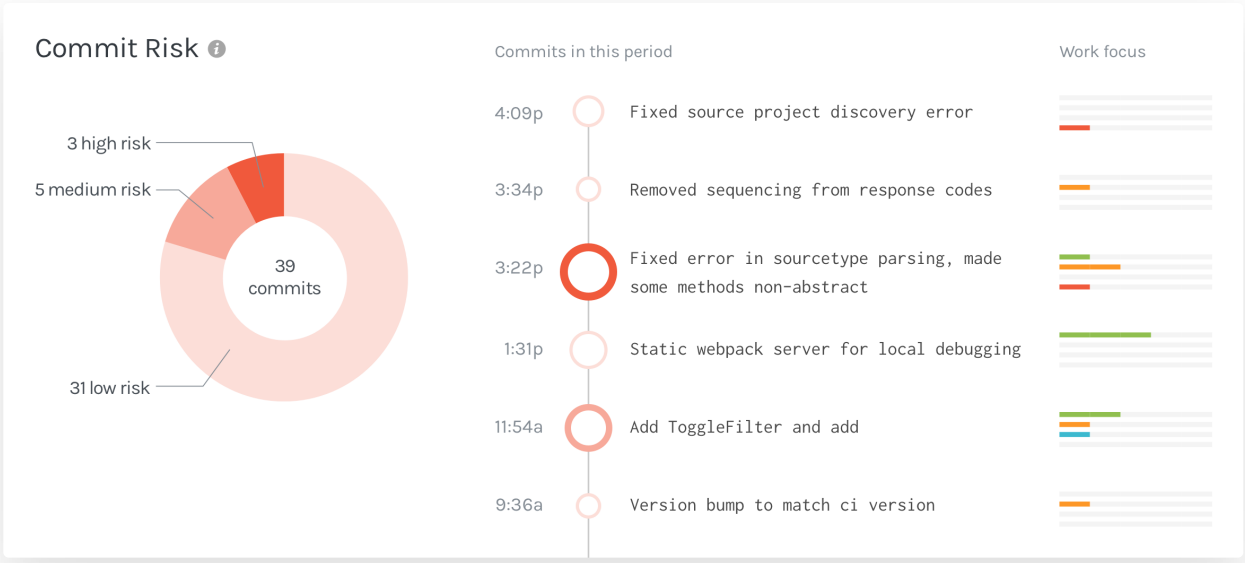
A team of 100 engineers may cost in excess of $10 million a year in fully loaded costs. So, documenting a 20% increase in output provides a path for engineering management to demonstrate multiple millions of dollars in value generated for their company.

Over time, this engineering team has steadily increased their Impact to the codebase. Whether this is a result of hiring additional engineers or improving team performance, this graph paints a picture of success that the rest of the organization can understand.

INSIGHT 02

Commit and Pull Request Behavior

Receiving visibility into commits and pull requests as they occur, as well as to their content, code profile and complexity, is a direct way for managers to better understand the progress, and challenges, of their team members.



Software engineering managers rely heavily on daily stand-ups and 1:1 check-ins to understand the progress of their engineers. The portion of this verbal interaction that is focused on diagnosing risk and getting status on progress can be reallocated to productive coding time if the team manager instead uses a data-driven dashboard that shows commit progress and PR reviews as they happen.

While viewing commits in a data dashboard may not sound sexy, it is an easy way to eliminate one to three hours of unnecessary meeting time each week for each team member, which can result in roughly a 5% increase in productive time.

A manager's time is zero-sum and should be applied for maximum impact. When reviewing code commits, data cuts through the noise and signals which work carries an elevated risk profile to be prioritized for additional review.

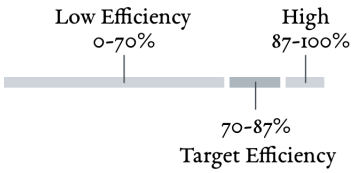
INSIGHT 03

Code Churn

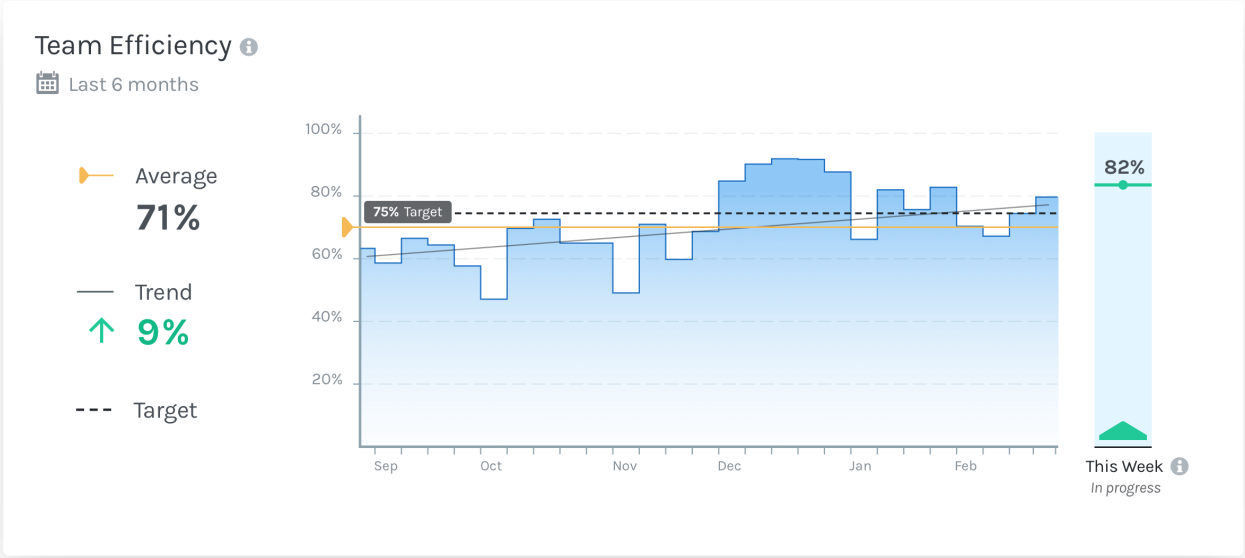
Code churn, or code rework, is not a bad thing. Testing and rework are natural parts of the software development process. However, code churn levels that deviate significantly higher or lower than expected norms can represent smoke that is an indicator of a potential fire.

In benchmarking the coding behavior of over 85,000 software engineers*, Pluralsight found that code churn levels most frequently run between 13-30% of all code committed (i.e. 70-87% Efficiency), where a typical team can expect to operate in the neighborhood of 75% Efficiency.

Since a baseline level of Code Churn is always expected, only when Efficiency moves materially above or below 75% should there be cause reason for concern—something is likely amiss. A churn level of less than 10% would indicate that an engineer is potentially sacrificing speed for precision; a churn level of over 25% would suggest that an engineer may be stuck, or is working on a project where they need assistance.



***The Fundamentals Metrics, a Pluralsight data science study of 7M commits, across 1.8M active days, from 87,000 authors which established statistical evidence for industry benchmarks and Pluralsight's Fundamentals and its associated Leadership Playbook.**



By baselining the 'natural' churn levels of typical types of projects, engineering managers can actively monitor churn by engineer, or by project, to identify areas where their team may be hung up and need assistance. A particularly high (or low) churn level could be an early indicator that a project is not progressing as planned.

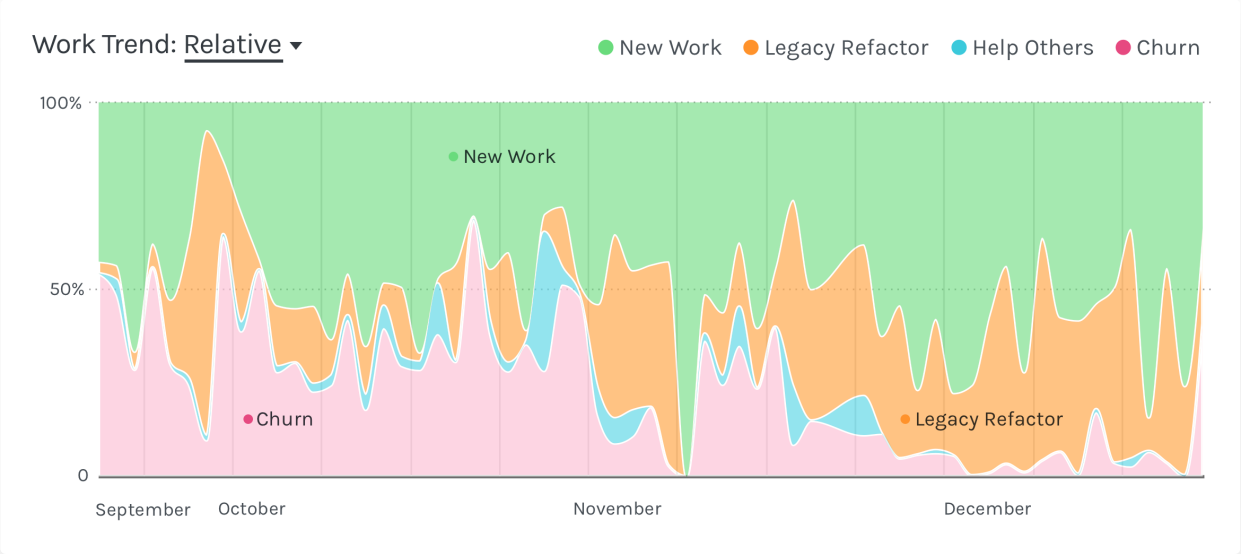
Over the past six months, this engineering team set a target of 75% Efficiency and is now working within industry norms.

INSIGHT 04

Legacy Refactor vs New Work

A common question that is posed to both the CFO and CTO is 'How much of our software engineering investment is spent on new work, versus supporting or refactoring legacy code?'

This is a metric that can be easily quantified by analyzing code at the time of commit, creating hard data on how much of a team's productivity is dedicated to new projects versus to technical debt.



It is now possible to quantify the percentage of work delivered that relates to legacy refactoring down to the line level. A 100-person engineering team that spends 30% of its time on legacy refactor is spending over $3 million a year working on older code.

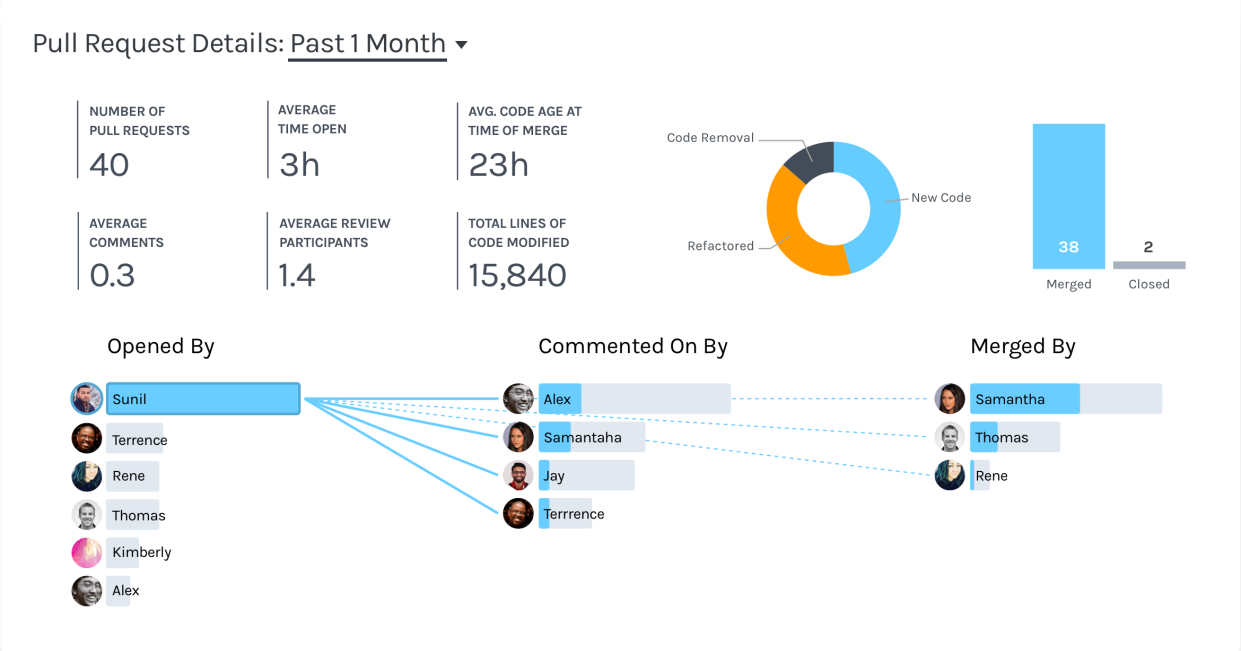
Incorporating this data into the standard KPI set of a software engineering organization should be a no-brainer, once the values are accessed and aggregated from the source code repositories.

At the beginning of the period, this team focused on new and exploratory work as indicated by the New Work (green) and prominent Code Churn (red). In mid-November, they transitioned to paying down technical debt with an emphasis on Legacy Refactoring (orange).

INSIGHT 05

Collaboration trends in code reviews

Rating Pull Requests by their underlying code complexity, and quantifying the number of reviewers and comments, is a simple way to establish metrics for the code review process.



This data is accessible in the code repository and provides the engineering manager with a clear input into the collaboration process that is taking place around code reviews, providing another example of how simple quantification of engineering behavior can make a manager more effective.

When too much time is spent in code review, it can be an organizational drain; if not enough review occurs on high complexity PRs, it can put the broader code base at risk. By using data, managers can optimize the time spent on code reviews while also decreasing downstream risk to the codebase.

In addition to better managing pull requests that are in process, managers can do systematic review of past PRs to identify healthy, and risky, collaboration patterns.

Data allows managers to identify collaboration trends like how long Pull request are staying open, if review protocols are being followed, or when an unusual amount of PRs are being rejected (closed).

WHAT'S NEXT

Will software engineering embrace metrics?

We are at an exciting threshold to the software engineering world entering the realm of data-driven management. Some key points to understand about adopting data-driven metrics within software engineering organizations:

Data does not replace management

Data is not a substitute for management—it is a tool that makes management better. By using data, managers can better understand risk, identify bottlenecks and replace low-value meetings with analysis of trends in the codebase.

Data must be used for good

The cultural approach taken to embracing and utilizing metrics is critical in determining the effectiveness of a data-driven management approach. Everyone involved must agree that the purpose of using metrics is to make the whole team better. The emphasis needs to be on team improvement and learning so that everyone can get better and create a better work product.

If it is perceived that metrics will be used punitively, arbitrarily or without context, then it is possible to create a defensive or destructive culture. From the top-down, the emphasis should be on improvement, growth and increased self-awareness so that the whole organization can evolve.

FLOW
POWERED BY GitPrime

With the 2019 acquisition of GitPrime, Pluralsight Flow gives you the confidence you need to accelerate velocity and visibility into and across your software engineering teams.

01. Engineering leaders have been operating in the dark.

For many organizations, software engineering is one the most expensive and mission-critical departments. Companies invest millions of dollars in software engineering without a feedback loop to understand how well we're doing or where to focus on improvement.

02. Flow turns the lights on with objective data.

Flow generates actionable metrics to optimize release processes, improve collaboration workflows and remove bottlenecks, while creating unprecedented visibility for all levels of management.

03. Get deep visibility into your development process.

Flow instruments the tools in your development workflow—from commit data, pull requests, tickets, and more—to provide actionable insight into individual and team workflows.

04. Turn workflow data into operational improvement.

Flow gives software leaders a fact-based view of effectiveness and performance—with prescriptive metrics to drive process improvement. The end result is improved quality, more time spent coding, healthier distribution of knowledge, and faster time to market.

About Pluralsight

Pluralsight gives you confidence you have the skills and insights you need to execute your technology strategy. You can upskill your teams into modern tech roles, improve workflow efficiency and build reliable, secure products. We are the technology skills platform.

By leveraging our Skills product, which includes expert courses, skill assessments and one-of-a-kind skills and role analytics, you can keep up with the pace of change, put the right people on the right projects and boost productivity. With our Flow product, you can debug your development processes with objective data, identify bottlenecks and keep a pulse on the health of your software teams.

Used together, they empower you to develop, measure and deploy critical skills at scale and improve engineering effectiveness.

Visit pluralsight.com/business to learn more