PLUTORA

EGUIDE

The Development Battle Nobody Talks About

(Achieving DevOps at Enterprise Scale)

Introduction

Who Should Read This White Paper

This white paper is targeted at technology leaders who have adopted a DevOps program but may be experiencing any of the following:

- Feeling you're **missing something** to bring it all together
- Lacking the level of **business involvement** you expected
- Seeing technology improvements, but not driving the **business**

What You'll Learn

In this white paper, you'll learn how DevOps is the start, but not the end, of a successful collaboration between Dev and Ops and that it has little to say about an effective collaboration between the business and technology.

Through an exploration of the history of DevOps, you'll learn that **release management** is the last piece in the puzzle for connecting Dev and Ops and that **value stream management** is the overarching component to have your DevOps program deliver consistent and meaningful business outcomes.

Technology and Business

The Goal of Technology in Business

No matter the industry, company size, or even type, technology exists to support the business in realizing its strategy and achieving its goals. Technology isn't there for its own sake, but to deliver outcomes for the business. This is the most commonly misunderstood aspect of technology leadership and also the most critical. There are a variety of benefits your business colleagues expect to receive from technology, some of which include the below:

Any business strategy requires technology delivery that yields one or more of the above benefits. On the surface, most of these benefits would seem to derive from technology simply acting more efficiently. For example, for technology delivery costs to be low or for new features to get to customers more quickly,

Why isn't it enough for technology to just be crisper at execution? As we'll see, technology efficiency is both challenging to coordinate in and of itself and also not likely, on its own, to lead you to the business outcomes you want.

Get new functionality to market quickly and cheaply

Deliver functionality that is of high quality

Exceed your customers' needs

Enable superior customer support or service

Keep technology costs low

Prior Attempts at Aligning Business and Technology

The first attempt at aligning business and technology was through the **waterfall methodology**. Waterfall is out of favor now (outside of a few specific industries), but in theory, it makes a lot of sense, particularly if you believe that technology efficiency leads to business outcomes:

- Shouldn't the business be able to tell you exactly what they want?
- Shouldn't it be maximally efficient to wait until the business can say what they want?
- Isn't it guaranteed that if we meet the requirements, the project was a success?
- Isn't the delivery of the project a guarantee of delivering the business outcome?

Of course, now we know that the last question is the most important, and the answer is no, delivering the requirements you gathered six, nine, or 12 months ago doesn't guarantee your desired business outcome. As such, even if it's more efficient for technology, it's riskier for the business.

The failures of waterfall led to **agile**, which made the opposite assumptions: business and technology staff should work together every day, and changes should be embraced, even late in the development process. Most firms today at least claim to be agile (to varying degrees), but there are still several key challenges that must be overcome:

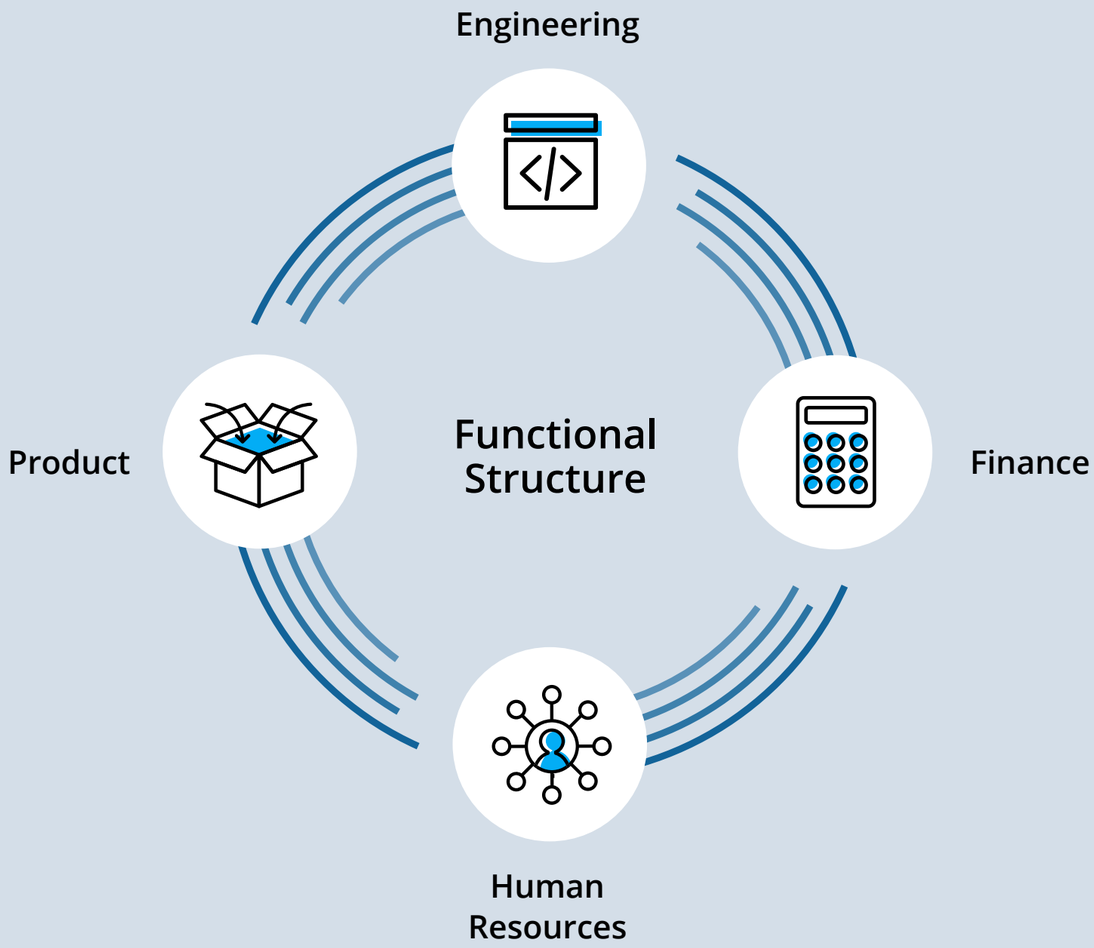
- How can you best align yearly budgeting and planning with a paradigm that intends to support late changes?
- How does agile make technology better at doing its work?
- How much does the business even care about the inner workings of an agile framework like Scrum?

The Importance of Aligning Technology With Itself

It’s one thing for a company to experience challenges in aligning across departments, but surely technology is able to align itself to maximize delivery, right?

That’s easier said than done, and it tends to begin with the design of the technology organization. Generally, companies are structured functionally, such as product, HR, finance, and engineering being separate departments.

The rationale in such a structure is that domain-specific expertise (and thus efficiency) can be derived and this cascades down. In other words, an engineering or information technology unit would often end up with Development and Operations departments. It’s at that point where a commitment is made to functional excellence that the seeds of conflict are sown.



Development vs. Operations

The Historic Conflict

Here's how this conflict typically unfolds:

- **Leadership pressures developers** to put out new features as fast as possible.
- **Developers work fast** to meet prescheduled release dates.
- **Developers hand software over to Operations** to deploy, operate, and support.
- **When incidents occur**, **Operations is on the hook** for support tickets, late-night phone calls, and high-priority incidents.

To compensate, Operations introduces some methodology for controlling changes and releases to maintain **environment stability**. Since information security practices recommend separation of duties, Operations is in the driver's seat on releases: they have the admin access and credentials to make production changes, and developers must play their game to have their changes released.

Unsurprisingly, stakeholders and firm leaders see this lack of progress and they push harder, which tends to make quality slip more, and the race is on.

Ideally, Development and Operations would work together to achieve solutions that are the right trade-off of speed and stability, but the problem isn't the people—it's their incentives.

The Impact of Incentives

Organizations use goals to drive progress. This is a rather obvious statement, but the implications are important. A technology department will typically contain distinct subgroups, such as Development, Operations, and governance. The head of the technology department is accountable for all technology goals, and it's fairly easy to take those goals and split them up among the subgroups. In so doing, functionality throughput goals (like make X progress on Y project) are given to Development, stability goals (business systems must be up 99.99% of the time, or there must not be more than one incident of a high priority) are given to Operations, and budget goals are given to governance.

Here's the issue: there's no incentive within the system for any of those three groups to do anything but focus intensely on their own goal. Even if it's better for technology (and the business) overall to release more functionality, staff in Operations won't typically see any performance benefits (e.g., a bonus) to doing so.

The Gulf Between Dev and Ops

The mismatch in incentives, speed vs. stability, is the source of the conflict between Development and Operations. Historically, cross-functional teams are seen as the solution to these sorts of conflicts. Simply creating a team composed of Development and Operations colleagues, it's believed, will magically resolve this tension and lead to higher degrees of both speed and stability. And in truth, cross-functional teams are a step in the right direction and will tend to lead to positive conflict and better navigated trade-offs.

But look at what's still unchanged. First, the dynamics between the two departments are still not resolved. Development and Operations, in aggregate, are still not incentivized together. As a result, while members of a cross-functional team will forge a shared identity, colleagues outside of those teams, still measured on the old incentives, will not, and friction will still exist. Second, the business doesn't really want speed and stability to exist in a trade-off at all. Instead, they want the seemingly impossible: more speed and more stability.

The Rise of DevOps

How DevOps Rose to Prominence

DevOps rose to prominence in acknowledgement of the long-running, misaligned incentives between Dev and Ops and the high costs that such was imposing on the business. From the business's perspective, they want to get more value per dollar out of their strategic technology investments, viewed as new functionality, and they don't really want to pay more for stability. In addition, agile began to cede ground to lean, and lean heavily drove the key idea of working in a flow-based way and avoiding batches and works in progress. The separation between Dev and Ops can be broken down to DevOps not through cross-functional teams, but by rethinking the entire relationship between Development, Operations, and the business entirely.

Tooling and Culture Impacts

DevOps puts pressure, in a positive way, on your tooling and culture because the respective Development and Operations methodologies still impose constraints that must be satisfied. On the Development side, there's all the usual tools for writing, building, and deploying software. On the Operations side, there are still requirements to understand and manage changes going to production in a consistent, repeatable way. To be a successful DevOps team, your entire toolchain from front to back must be examined and brought in-line with these needs:

- How developers manage **source code changes** among the team
- How source code changes are applied to a **target preproduction environment**
- How new features are tested in a preproduction environment against some list of **acceptance criteria**
- How new functionality is deemed suitable for **deployment to production**
- How functionality is actually deployed to production and **released to customers**
- How updated systems are monitored to **ensure stability**

Work-item tracking systems and automated build systems are typically used in tandem to meet many of these needs:

- At source code check-in, **automated jobs** build the software and run tests.
- If the tests pass, the **code is deployed** to a target environment.
- **Additional tests** (manual or automated) are now run against the software in its environment.
- The software progresses toward production, with an **increasing amount of confidence** as to its state from a progression of automated tests and checks run against it.

Key Operating Improvements

A company executing a successful DevOps program will tend to have the following characteristics:

- **Less time** between code being changed and getting to a customer Deployments occurring **more often**
- Less time to **restore a system** after an incident
- **Lower frequency** of a deployment failing and requiring resolution

These metrics also can serve as a forcing function for DevOps adoption: given a new DevOps team, focus their continuous improvement efforts on these metrics and let them drive their own improvements.

What DevOps Demands From You

Successful DevOps adoption demands that you overhaul your technology processes, structure, and goals to ensure that both Development and Operations have a vested interest in each other. Instead of structuring both groups for local optimization, you instead structure them for better outcomes at the technology team level. This goes far beyond standard cross-functional teams and into the design of your technology organization itself.

DevOps Is Not the Silver Bullet

If we can get all these great benefits from DevOps, why isn't that enough? If DevOps can get us to speed and stability, won't that, by induction, result in better outcomes for the business? On the one hand, yes, absolutely, the business will benefit from these changes. On the other hand, though, we still haven't materially upgraded the nature of the relationship between business and technology. As such, we haven't improved the alignment between the two teams, and we're still undercutting our ability to deliver meaningful business outcomes.

The New Challenge of DevOps

Why Didn't DevOps Fix Everything?

DevOps didn't fix everything because DevOps is fundamentally a set of technical practices. Adoption of DevOps leads to running a better technology group, but by intent, it never set out to address business and technology alignment. As such, DevOps is a great thing to do, but most business leaders are unlikely to get that excited about it, for the same reason they were initially optimistic about agile before the bloom fell from that rose: to them it just looked like a lot of technology changes they didn't understand and that didn't seem tied to what they do care about.

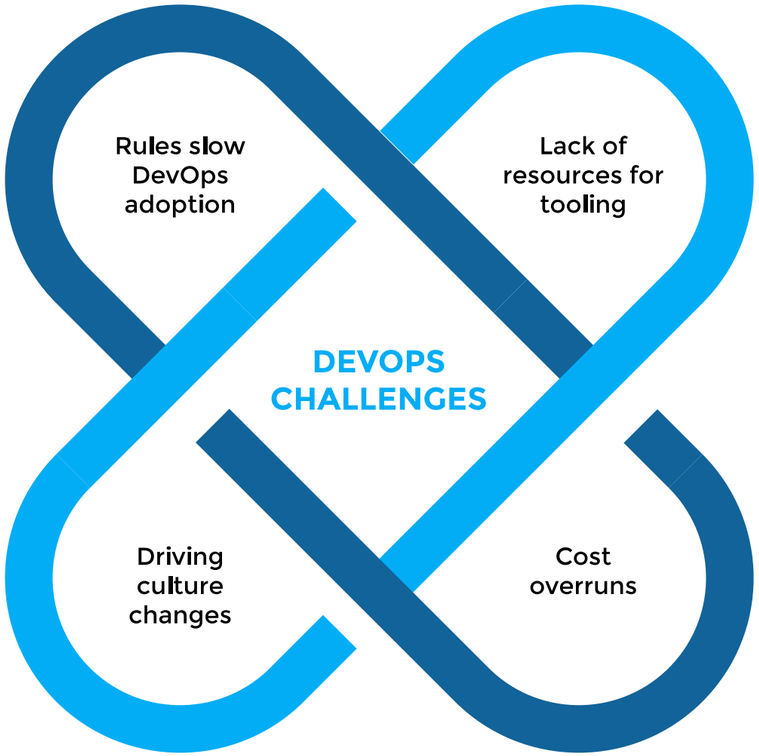
Can You Be Too Big for DevOps?

In general, no, but there are DevOps challenges unique to firm size.

For small firms, there tends to be a lack of resources to build or buy the necessary tooling.

For larger firms, the tooling is normally less of an issue, but two other issues may exist. First, larger firms tend to accrue a lot of rules and procedures. These rules slow DevOps adoption because you must gain the support of the rule owners for modifying the approach (typically automating it within the DevOps pipeline). Second, driving the necessary culture changes are harder simply because there are many more people that need to internalize and carry the message for you.

Regardless of firm sizes, most firms underestimate the level of time and effort that are necessary and either end up in a half-evolved state or, if they do get to the finish line, the ultimate benefits don't overcome the perception that the cost overruns blew away all the value.



Can the Issues Be Fixed?

The issues can be fixed, but they won't be fixed through more technical practices. If it were possible for changes isolated to a technology group to actually drive meaningful improvements in business outcomes, it would've happened by now.

The solution, then, is to actually address the alignment between business and technology, to materially improve those collaborations in an ongoing, process-centric manner, and in so doing, finally solve this problem.

Value Stream Management for Better DevOps

The New Model for Managing Software Delivery

The superior model for the software delivery process is to not view releases as huge, big bang events that raise the stress of everyone involved. It's also not for the business to be told what's going to be included in any particular release as if they're an outsider with little vested stake in them.

DevOps adoption leads you to more frequent, smaller releases but says nothing about how a business stakeholder is intended to influence them. It's here that value stream management takes up as the superior model for software delivery in a DevOps world.

> A **value stream** is a model of how your product or service gets to a customer

The customer is at the heart of a value stream–focused approach. As the name suggests, this should be a smooth flow and not done in chunks, which aligns nicely with DevOps. The business enters the picture because it's their prioritization and collaboration with their value stream that gives them the control of how value gets to the customer.

Specifically, while DevOps helps make the process within technology more consistent, value stream management focuses on the system-wide approach of work from idea to production, not just to it reaching the development process. The latter of these is the benefit of value stream management and yields the final piece for consistent, predictable value delivery.

> **Value stream management** focuses on the system-wide approach of work from idea to production

Orchestration and Automation

Orchestration and automation demand a few things from your systems:

- The schedule or calendar for releases must be publicly available to all stakeholders.
- The criteria for how and when items are releasable must be published to all stakeholders.
- The status of any item and its progress to release must be publicly available to all stakeholders.
- The release and deployment pipelines must be as automated as possible, subject to the compliance and governance requirements you have that necessitate manual checks.

Governance, Compliance, and Audit Considerations

Within a DevOps world, there are still requirements related to compliance, audit, and governance that must be satisfied. A few specific ones that occur often and that have implications for the software delivery process:

- **Payment Card Industry Data Security Standard (PCI DSS)** is in play if you handle credit card data and it affects any system involved in such processing.
- **The European Union's General Data Protection Regulation (GDPR)** may be required if you're interacting with data on EU residents.
- **Sarbanes-Oxley (SOX)** is a financial reporting requirement if you're a public company in the US—that has technical control implications.
- **ISO 27001** is an information security standard—change management is a key process that falls under its purview.
- **Control frameworks** that your company has chosen to adopt such as COBIT or CIS-20.

While every company's context is different, many times these can be done automatically if you collaborate with your internal stakeholders:

- The release/deployment/development pipeline changes automatically update logs to make available for audit.
- Security requirements are "shifted left" into the pipeline and automated so that security issues are caught early.
- Additional tests can be automated within the pipeline to satisfy compliance needs.

Value Stream Management

Value Stream Management Fundamentals

As mentioned above, a value stream is a model for how products or services reach a specific customer. In an absolute sense, the value stream is all activities that must be performed on some work item before it reaches the customer. **Value stream mapping** is typically one of the earliest activities in setting up value stream management that starts by identifying every single process that acts on an item in that chain. Usually, those initial value stream maps are horrifying with a tangle of activities that kick the item back and forth. And that's okay! It's more important in the mapping phase to be comprehensive. From there, optimizing the value stream is about two steps: cut out everything that doesn't add value and keep the item moving in one direction only.

Categorizing an activity as value or waste can be tricky. For example, is compliance waste? On the one hand, the customer won't directly experience compliance, so it's possible to argue that it is. On the other, however, the customer derives benefits from the compliance, even latently, such as confidence that the product or service meets some external standard or accreditation process. The point isn't that compliance is or isn't, simply that you must think very diligently and holistically about those activities you're doing and how they add value for the customer.

Aligning Business and Technology Through Value Stream Management

Team Structure and Workflow

A value stream will include team members from both the business and technology groups. Most commonly, technology supplies a cross-functional team, especially a DevOps team, with developers, testers, an Operations engineer, and a user interaction designer. The business supplies a value stream manager and typically one or more members to serve as product owners for the work items.

The value stream manager is the ultimate decision-maker and works with the product owners to prioritize and sequence work items. The Development team then collaborates closely with the product owner or value stream owner to understand and execute the item in a continuously flowing system. Ideally, items start from concept and proceed through the value chain and to the customer in production.

Resource Allocation

Like most efforts, there are two principal types of resources that will be needed. On the human side, both business and technology groups will supply team members for the value stream. On the finance side, the business group supplies the monetary resources to fund the value stream. For value streams, the financing occurs by funding a value stream for a period of time and evaluating it over time. Teams can also be adjusted as needed though borrowing from most agile perspectives and research on team formation; it's best to leave teams intact for a period of time.

Continuous Improvement of Value Streams

Value stream management is concerned with several aspects:

- Is the value stream delivering the right level of value based on the investment? In other words, are the target metrics being improved?
- Is the value stream operating as efficiently as it could be?
- Is the value stream team improving its own operations?
- Is the value stream operating within the organizational policies and constraints?

Effective value stream management is not a one-time goal that your organization can achieve and leave to run unattended. Like all processes in the software delivery cycle, it must be continuously evaluated and improved with a goal of deepening the partnership between the business and technology, and driving meaningful business outcomes.

Portfolio Measurement and Evaluation

Portfolios can now be viewed as a collection of value streams instead of a set of fixed projects.

Instead of waiting for projects to be completed and then measured, months after the fact, value streams lend themselves to continuous monitoring and evaluation as the more releases that occur, the more value is making it to customers.

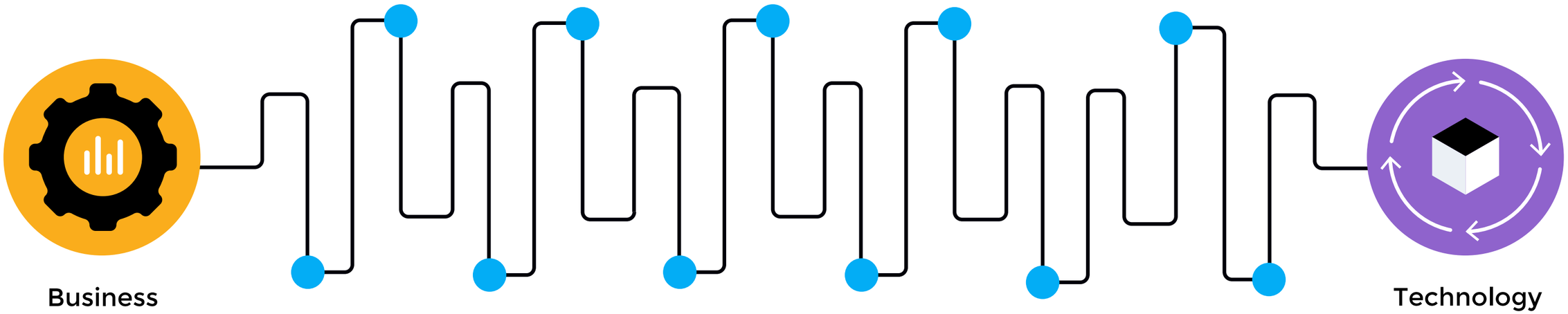
This puts senior leadership or portfolio managers in an even better position to add or move around resources (in the form of value stream funding) or to evaluate value stream leadership for their strategic choices.

A More Effective Business DevOps

The challenge inherent in any technology team is to demonstrate themselves to be a strategic partner and not just an order-taker. Internal technology team reorganizations, efficiency efforts, and a focus on technology's operating metrics won't get you there. Ditto with efforts to embed business partners with development teams or believing that crisp project execution is sufficient to win the day. These are all necessary—don't get me wrong—but the real secret sauce is speaking the language of the business.

Nothing will make technology a true strategic partner like aligning the business priority, via its resource allocations, directly to technology streams. That step gets you to the place you want to be: demonstrating, on an ongoing basis, that you're achieving real business outcomes in business terms.

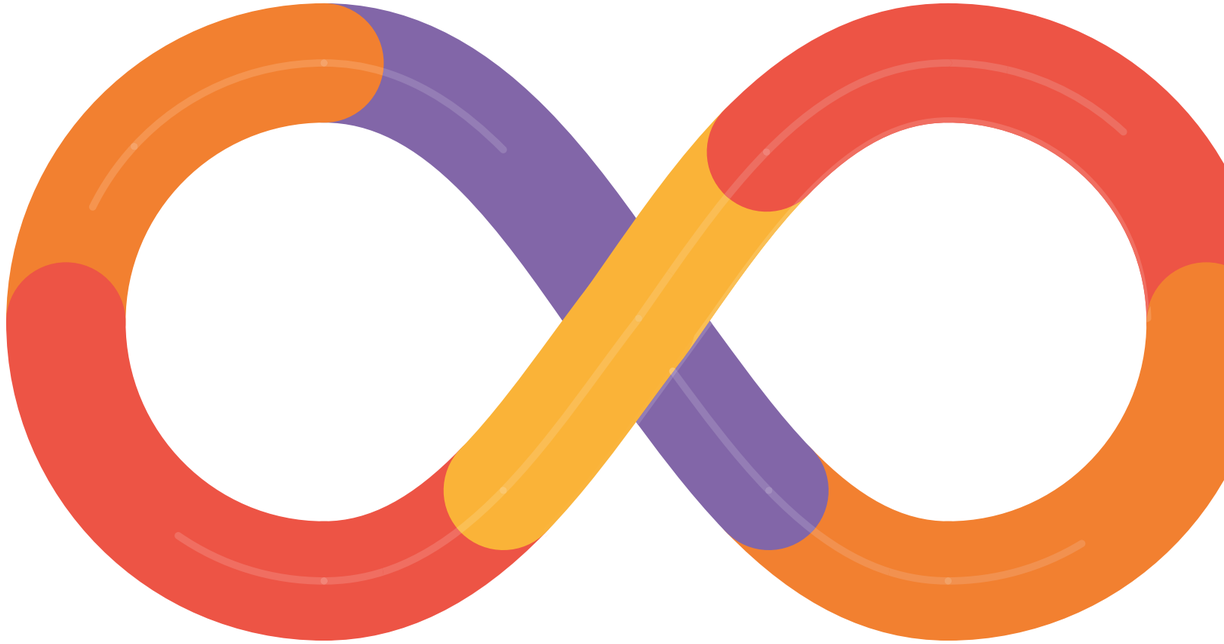
Value stream management is the link you need to take your DevOps program out of technology and attach it where the business will see and feel its value.



Wrap Up

DevOps is the start of a journey to drive better business outcomes, but it's not the end. You must continue this journey by ensuring you have rock-solid release management to unit Dev and Ops. Then, reinforce these technology successes through value stream management to align your business and technology around a continuous flow of customer value.

Plutora has the #1 value stream management platform to improve your software delivery process.



PLUTORA

Learn more: www.plutora.com
Email: contact@plutora.com

Plutora, the market leader of value stream management solutions for enterprise IT, improves the speed and quality of software creation by capturing, visualizing and analyzing critical indicators of every aspect of the delivery process. Plutora orchestrates release pipelines across a diverse ecosystem of development methodologies, manages hybrid test environments, correlates data from existing toolchains, and incorporates test metrics gathered at every step. The Plutora Platform ensures organizational alignment of software development with business strategy and provides visibility, analytics and a system of insights into the entire value stream, guiding continuous improvement through the measured outcomes of each effort.