

PLUTORA

The Evolution of the Release Manager

By Helen Beal



Who Should Read this Paper

This paper has been written for **release management professionals**, who are likely using ITSM based processes in traditionally operating information technology environments.

Its purpose is to lay out the vision for a DevOps led release process in a value stream management centric environment while acknowledging the myriad challenges in moving from one way of working to another.

A value stream is anything that delivers a product or service. Value stream management is primarily concerned with optimizing the flow of value from idea to realization in the hands of the customer.

It aims to provide usable methods proven by practitioners in the field to make these changes happen. We have interviewed several release management professionals whose advice is quoted in the paper.

Lead author: Helen Beal

Contributors: Nadeem Augustine, Tony Mongiovi, Simone Jo Moore, Mark Peters, Ryan Sheldrake

Reviewers: Scott White, Jeff Keyes



Table of Contents

Who Should Read this Paper	2
1.0 What Release Management is and Why it Exists	5
1.1 Why They Exist: Release Managers, Weekends and Calendars	8
1.1.1 Release Managers	8
1.1.2 Release Weekends	11
1.1.3 Release Calendars	12
2.0 What DevOps Says About Release Management	13
2.1 Long Term DevOps Vision	16
2.1.1 Establish a Common Vocabulary	17
2.1.2 Small Batch, Incremental Change Reduces Risk	18
2.1.3 Think and Act As Value Streams	19
2.1.4 Automate for Consistency, Predictability and Risk-Reduction	20
2.1.5 Architect for Small Batch Build, Test, Deploy, Release	21
3.0 The Reality of DevOps Evolution	22
3.1 Steps Towards DevOps Release Management	26
3.1.1 Reduce Batch Size	27
3.1.2 Architecture: From Monolith to Microservices	28
3.1.3 Transition the Release Management Role to the Value Stream Team	30
3.1.4 Increase Availability of Release Slots and Standardize Change	31
3.1.5 Automate the Release 'Checklist' and Deployment Process	32
3.1.6 Limit the Deployment Blast Radius: Dark Launches	33
3.1.7 Reducing the Route to Live, Leveraging Cloud	35

3.1.8 Keep an Eye on People, Behavior and Cultural Challenges	37
3.2 A Note on ITIL4	38
3.3 DevOps Release Management Best Practices	39
3.3.1 Connecting Planning	40
3.3.2 Data Driven Conversations	40
3.2 Use Cases for the Release Manager	42
3.2.1 Example Experiments	43
4.0 Release Management Automation	44
5.0 The DevOps Release Manager	47
5.1 Tips for Release Managers Wanting to Evolve	49
5.2 New Roles for Release Managers	53
5.2.1 Site Reliability Engineer (SRE)	53
5.2.2 Release Train Engineer (RTE)	54
5.2.3 Product Owner	55
5.2.4 Value Stream Architect	55
5.3 The Future of Release Management	56
5.3.1 Value Stream Management	56
5.3.2 AI and ML	57



1.0 What Release Management is and Why it Exists

ITSM defines release and deployment management as the process of managing planning and scheduling the rollout of IT services, updates and releases to the production environment.

The idea of a new software release originated when a new business case or a project was created by, for example, the marketing team. Traditional ITSM processes create release packages consisting of large bundles of features.

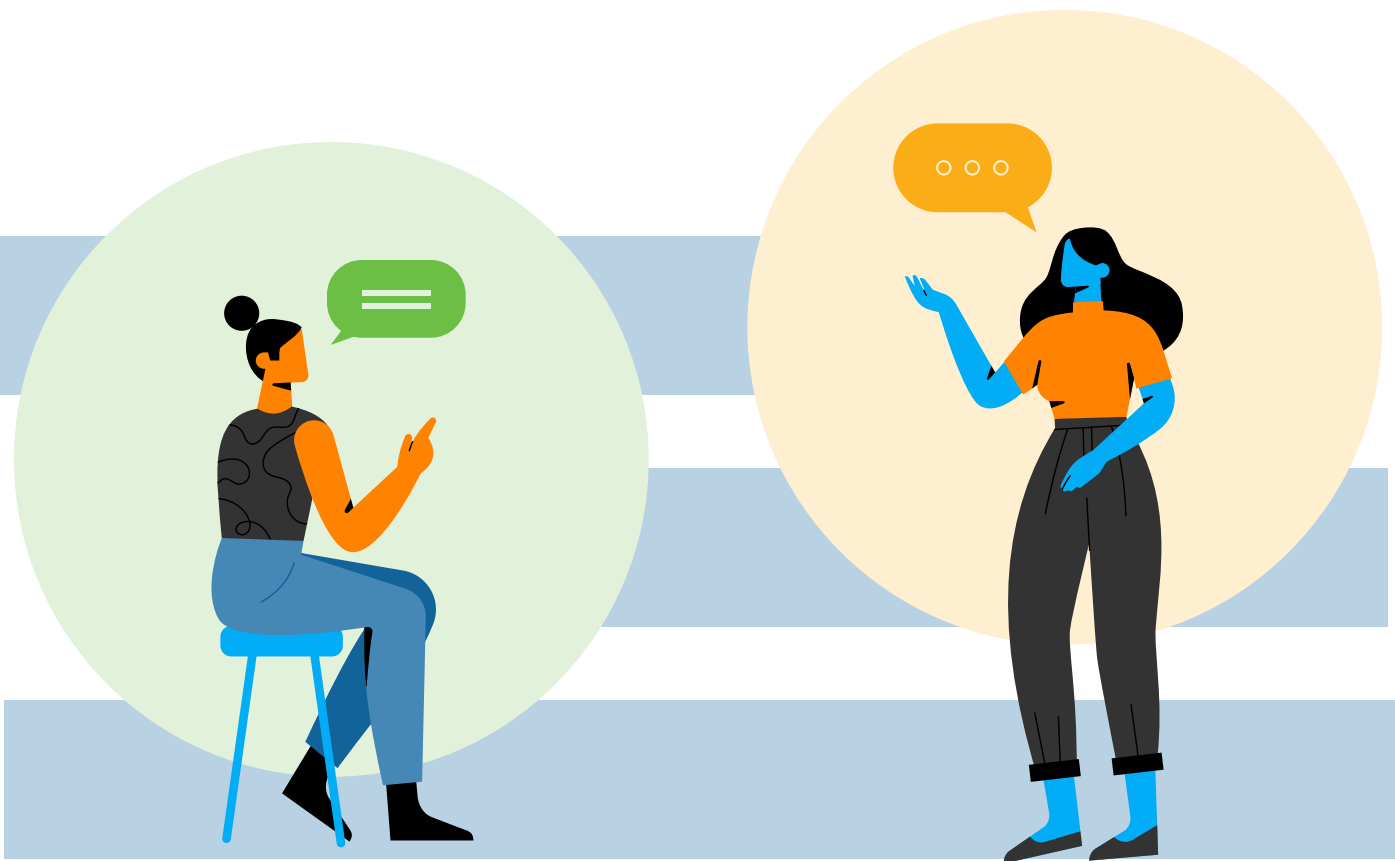
The hands-on work of release management has traditionally been the domain of those managing the IT services, the IT operations department. It was the release process where much of the conflict happened that triggered the DevOps movement.

This was because the development team created the release packages which they would then “throw over the wall (of confusion)” to the IT operations team to deploy. As development teams increased their release cadence as a consequence of agile adoption, IT operations were not ready. Problems would then occur as a result of a number of factors:

- The IT operations team **not understanding the release package**
- **Inconsistency** between pre-production and production environments
- **Poor quality code**; with so many parts it was difficult to pinpoint the root cause
- **Integration issues** with complex dependencies in tightly coupled systems

The conflict that arose from these painful release events manifested in several ways:

- The 'war-rooms' that gathered when inevitable outages occurred were characterized by **cross-team blame without evidence**
- **Development teams complaining** IT operations took too long to provision environments, or make test environments available
- IT operations becoming frustrated with **developers asking for administrator access** to production systems which was in contravention of compliance requirements (segregation or separation of duties)



1.1 Why They Exist: Release Managers, Weekends and Calendars

When things go wrong, our first reaction has traditionally been to create a control to try to avoid a repeat of the problem in the future.

This has served us well in many ways in mitigating the risk of making a change, but it's come at a cost, by introducing layers of bureaucracy and wait times.

1.1.1 Release Managers

All the paperwork; the calendars, the checklists, also the change advisory or approval boards (CABs) and all the hoops the teams in a value stream needed to get through to get the new features the business was pressuring them for meant specific roles were created mainly to manage these processes.

Change management and project management were involved, and increasingly release managers were hired. Sometimes these roles proliferated so there were release management teams. They were gatekeepers to going live.

Simone Jo Moore, a transformation consultant with many years of hands-on ITSM experience described it as:

“They were a barrier, the knights protecting the castle against the storming of the bastion by the development teams. There was a melee of forces fighting against each other. The customers were the crying peasants in the weeds of the castle begging for scraps.”

- Simone Jo Moore, Transformation Consultant

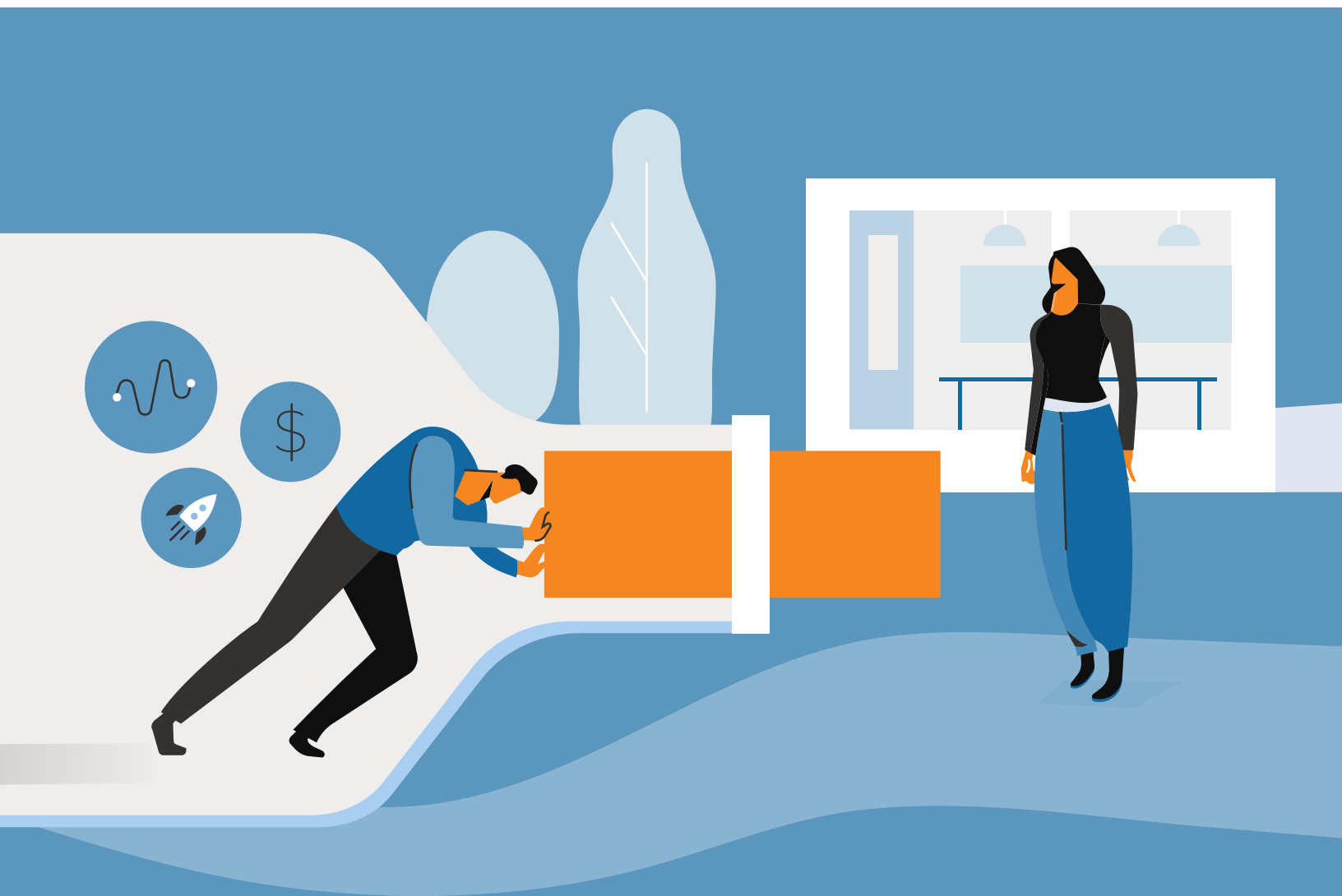
The DevOps Handbook, (p.63) recognizes release managers as a key role in supporting the value stream and describes them as: “the people responsible for managing and coordinating the production deployment and release processes.” Nadeem Augustine, who leads DevOps and release management for a global insurer, said:

“The release manager was a key role coordinating development, testing and IT operations. They’d combine all the steps, scope, phases and create a deployment plan. That would typically take three months or more. They were bringing the project together.”

- Nadeem Augustine, DevOps & Release Lead

As such, it's easy to see they, and the processes they are there to support, are a bottleneck. But it's also easy to understand why this **governance**, protecting the organization from disaster, is there.

The question then becomes how to remove this bottleneck so that value delivery can be increased and accelerated, without compromising the risk mitigation this model brought.



1.1.2 Release Weekends

The working environment for technologists has not been easy. As systems became more complex and interdependent, they became increasingly fragile. This, along with time constraints building technical debt in many forms, including that of systems becoming increasingly 'legacy' (or not using the latest, most advanced available technology), results in unplanned work.

Since releases were so big, and so unpredictable, teams were asked to work out-of-hours (for release nights as well as release weekends) to be on hand for when things inevitably went wrong.

These painful patterns of working lead to burnout. Ryan Sheldrake, who worked for many years in release and environment management roles in banking and retail described the environment as:

"It was like herding cats; trying to coordinate many teams in many locations to do things in an orchestrated manner. Release time was stressful - a crescendo of all these pieces coming together. A "this is it!" moment."

- Ryan Sheldrake, Release & Environment Manager



1.1.3 Release Calendars

Since these large, complex and interdependent releases were likely to cause issues and require intense and immediate remediation, they needed to be scheduled. Much of a release manager's job became scheduling these releases so there was visibility into what was happening when. Ensuring the completion of checklists or **runbooks** of tasks that could make release success more likely are also the domain of these release managers.

This is a control mechanism for managing those team and system interdependencies, but it slows every team's release cycle down as they battle to get on the calendar and wait for the slots to become available.

"You couldn't just trust that 'X' had done their job. There could be situations where the only person knowing what their job was the person doing it. Oftentimes, people didn't really appreciate the risk of deployment, the risk to the business. I was there to protect the service. Not only there was the risk of deployment failure, if we missed the maintenance window, requesting a longer one could mean a hefty fine as part of the customer SLA or from the regulator."

- Ryan Sheldrake, Release & Environment Manager

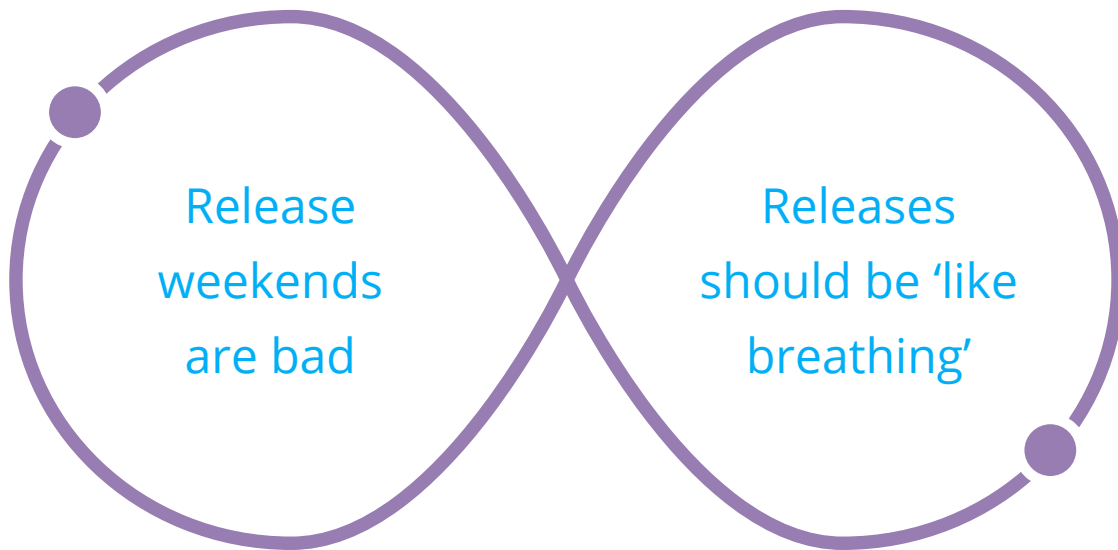


2.0 What DevOps Says About Release Management

Much of the pain associated with releases in traditionally operating environments has been caused by the separation of the development and IT operations teams. What is often referred to as a 'wall of confusion' exists between these two silos.

The DevOps promise is to break down these silos to balance throughput and stability so we can deliver more value faster and more safely.

Start with these two key principles in DevOps release management:

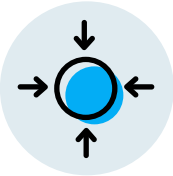


The need for these principles becomes particularly apparent when the development team 'throws new code over the wall' where it is caught by the operations team who now have to perform the release with little knowledge of its provenance.

This can cause many issues when things go wrong:

- Failure can be catastrophic
- It's extremely stressful for everyone involved
- It can be very expensive - not just to fix but in terms of fines and reputational damage
- It can cost people their jobs and/or progress in their careers
- It creates immense bad feeling between teams in contention as a 'blame-game' ensues

Moore identified her top 4 characteristics of release management in DevOps ways of working as:



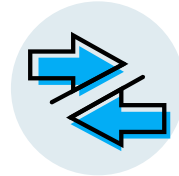
Smaller



Faster



Safer



Frictionless

“The conversations between development and IT operations need to be shifted and improved. Development making sure that IT operations know what they need to know continues to be a failure. Not including IT operations in design is a mistake.”

- Simone Jo Moore, Transformation Consultant

Agile and DevOps teach us to work in smaller increments to get faster feedback. This reduces risk and accelerates value delivery. As Moore identifies, this way of working is safer and reduces friction as we reduce handoffs between teams.

By reducing the size of the release event and making it safer, we can perform **continuous delivery** and no longer need ‘release weekends’.

Since we release small pieces so frequently, it's 'like breathing'; something we do so regularly it's no longer stressful. And so, agile and DevOps aim to create more sustainable working patterns and reduce the burnout seen in the traditional ways of working. Let's look at them in more detail.

2.1 Long Term DevOps Vision

In an agile and DevOps world, our teams can release their new features and fixes whenever they are ready - which is frequently and on demand. They build it, they own it. They are a multifunctional, autonomous team. They are long-lived and associated with a product, platform or value stream; the concept of project has gone.

Their continuous delivery pipeline ensures that software is always in a releasable state and they may also have continuous deployment - on successful completion of all the tests, the change is automatically deployed and released into production.

This vision sounds fantastic, but to get from the traditional state described in Section 1, to here, is not an easy road. There are a lot of humans working every day in ways they have been for many years that we are asking to unlearn and then learn these new ways of

working. Here are some steps that other organizations have taken during transition:

2.1.1 Establish a Common Vocabulary

For many organizations, it's important to ensure everyone understands what is meant by 'deploy' and 'release' as they vary frequently. Often, people prepare a release, deploy it to production and then release it to customers (some process frameworks, such as **SAFe**, specifically decouple deploy from release for this reason).

These distinctions become less important as DevOps fluency improves, but as teams and organizations evolve, they need to know they are speaking the same language by establishing a **common vocabulary**.

This new vocabulary will grow as organizations adopt DevOps principles and practices, so it's important that the DevOps journey is also seen as adopting a dynamic learning capability. Ensuring people share common understanding of what these techniques mean in their environments is essential.

2.1.2 Small Batch, Incremental Change Reduces Risk

Having large bundles of features (or 'batches') is in contention with the DevOps principle of little and often (incremental) where we reduce the risk of deploying a change by making it smaller.

Because we have large, high-risk releases in traditional ways of working we have to schedule them and have people to manage these schedules. Teams have to wait until their slots in the calendar become available in order to perform their deployment to production and release to customers.

As the authors of '**Accelerate**' wrote, when identifying deployment frequency as a key metric for measuring DevOps and organizational performance:

"Reducing batch size is another central element of the lean paradigm - indeed, it was one of the keys to the success of the Toyota production system. Reducing batch sizes reduces cycle times and variability in flow, accelerates feedback, reduces risk and overhead, improves efficiency, increases motivation and urgency, and reduces costs and schedule growth."

- '**Accelerate**' by Forsgren, Humble, & Kim

The authors identify deployment frequency as a proxy metric for batch size due to its low variability and ease of measurement.

2.1.3 Think and Act As Value Streams

Key to accelerating the flow of value from idea to realization is the destruction of silo-based ways of working. Having an **autonomous, multi-functional team** dedicated to a product or service (value stream) means we have all the people needed to orchestrate the flow of value from idea to realization in the customers' hands. Once we have identified the value stream and the team supporting it, it's time to map it.

Value stream mapping is an essential part of any journey so that the team can understand the flow of value (the cycle time) and how to make improvements. It's time consuming and intensive to do though, as a human-driven activity, so it's recommended that teams use value stream management software to automate their value stream map for continuous inspection and adaptation.

The recommendation is to think of the organization in terms of value streams and then allow each value stream to autonomously self-discover improvements and measure their own progress. As they make improvements they should continuously align to the high-level vision shared across all levels of the organization.

2.1.4 Automate for Consistency, Predictability and Risk-Reduction

The emergence of the **CICD pipeline** provided teams with hitherto unavailable capabilities to test earlier and fail earlier. At the same time, **orchestration tools** provided consistency in environment provisioning with cloud technologies.

This automation means releases become consistent as they are templated, predictable since we've done them before and experienced the outcomes; and this reduces risk.

Having a CICD pipeline, or better yet, an end-to-end **DevOps toolchain** that automates the entire value stream and provides traceability, continuous feedback and continuous compliance, means value stream teams have a view into every step of the value journey. This automation also means comprehensive testing has been performed early so we have confidence at the moment of a release.

Insights into the DevOps toolchain can be hard to extract though, potentially requiring manual interventions or the building of custom dashboards to get at the data. This can be overcome with the implementation of a value stream management platform. It connects all parts of the DevOps toolchain and therefore unifies the value stream. This allows the value flow to be continually inspected, insights to be gained and adaptations to be made to accelerate flow.

2.1.5 Architect for Small Batch Build, Test, Deploy, Release

Part of the issue in traditional enterprises is those monolithic systems with dependencies. When systems are tightly coupled, we have to build, test, deploy and release the whole thing every time. When we architect loosely coupled systems, they become componentized (e.g. **microservices**) and then it becomes possible to fragment the work batch size.



3.0 The Reality of DevOps Evolution

It's essential to recognize that changing the ways of working in any organization takes time. It's a human-driven endeavor and the journey is different for everyone.

The adoption of DevOps practices and behaviors should be approached as **incremental evolution**.

DevOps is underpinned by The Three Ways:

- 1. The First Way:** Optimize flow and seek a profound understanding of the work as it flows from left to right (system thinking/value stream centric ways of working)
- 2. The Second Way:** Shorten and amplify feedback loops
- 3. The Third Way:** Experiment and learn continuously

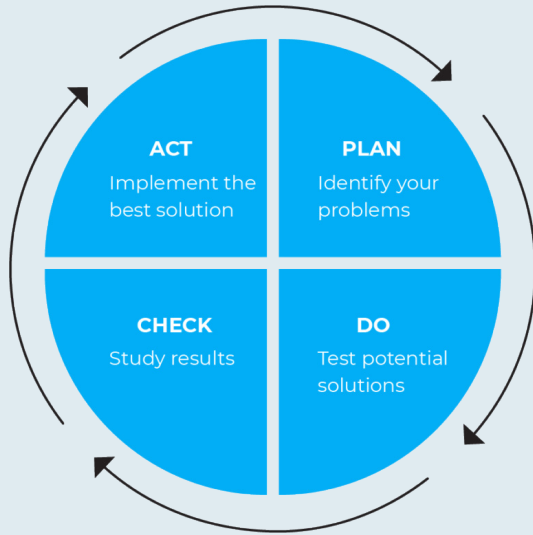
Many organizations have false starts, or drive some change forward only to see it falter, perhaps due to a change in leadership, or in market conditions. Then the momentum builds and the evolution accelerates again. When there is sufficient desire from enough people, the shared vision can provide goals and daily work can be reconfigured to allow room for the investment in the future state but there are always barriers that change agents will face:

Cultural and process debt ("This is the way we've always done it.")

- Leadership unwilling to distribute authority
- Lack of trust between teams
- Missing system thinking/value stream centric ways of working

Technical debt ("We can't find the time to save time.")

- Tightly coupled monolithic systems
- Lack of automation (version control, testing, environments)
- Missing access to cloud technologies



These barriers are hugely challenging and complex, but can and have been addressed in many organizations. Underpinning the evolutionary approach is the concept of making improvement a daily habit using the improvement kata where we hold the long term vision and goals in our minds and continually experiment (plan > do > check > act (**PDCA**)) between the current and next target states.

“In most organizations, there’s a conservative guard who mistrusts new ways of working. They don’t truly believe it’s safer. Governance and guidelines are still required - but doesn’t have to be the big hard stick people think it is - the guide rails need to be built in all the way through (continuous compliance).”

- Simone Jo Moore, Transformation Consultant

There are a lot of moving parts in a DevOps evolution and an entire organization made up of several hundred, thousands or tens of thousands cannot change overnight. We simply don't have the cognitive load and, of course, there's all the work that needs to be done to keep the business we have running. We hear phrases such as, "We're sprinting just to stay still."

For a release manager, this means distributing authority and evolving their own role from centralized command to an enabler who understands that different value streams have different capabilities and needs. Mark Peters, a cybersecurity expert and DevOps leader benefiting from years of operational experience in the military, described his top attributes of release management in the context of DevOps ways of working as:

1. "Automated processes
2. Distributed accountability
3. Accelerated delivery"

As previously mentioned, DevOps recommends we use an improvement kata in our daily work as an improvement habit. It has four, repeated steps:

1. Set the long term vision
2. Understand the current state
3. Identify the next target state
4. Plan>Do>Check>Act (PDCA) between the states

The PDCA cycle is about experimentation. The best experiments are empirical, that is data-led, and date specific. Using a value stream management platform to provide the data to support the experiments results in data-driven conversations about real and measurable progress.

This section is about the kind of experiments organizations can use to improve their release management processes using DevOps principles and achieve those goals.

3.1 Steps Towards DevOps Release Management

Value stream mapping is likely to show that in traditional ways of working the release and deployment process is lengthy. This is particularly true if teams are required to interact with a release management team to book slots on a release calendar.

Despite its insights value stream mapping has some downsides:

- It's human-intensive
- It's hard to coordinate
- As a result, it's rarely repeated

Value stream management platforms can easily extract the four key **DevOps metrics** for teams in real-time to inspect and adapt accordingly.

But there are some limitations to using these four metrics only:

- **Lead time** is defined as time from code commit to live - in value stream management we also care about time from idea or customer request to value realization
- **Deployment frequency** is interesting when teams deploy infrequently but are improving, but when teams achieve the capability of releasing on demand, other metrics become more useful

Because of this teams need to adjust metrics to team capability over time. This is one reason it's so important to **empower value stream teams to self-discover their own metrics** and measure them themselves. Another reason is that change is more effective when people are doing it for themselves, not having it done to them.

3.1.1 Reduce Batch Size

Reducing the batch size reduces the risk and the queueing time (the time an activity is waiting because it's in a queue). When changes are small they can be deployed easily with reduced risk of disruption and the distinction between the terms becomes less important.

Whilst some key metrics can be accessed through individual tools such as backlog or release automation, it's harder to gain insights into overall queueing time without a value stream management platform.

Unifying all parts of the DevOps toolchain automates the value stream map, creating a single system from idea to realization that enables teams to understand their cycle time and value flow efficiency so that they can see how changes are improving work.

3.1.2 Architecture: From Monolith to Microservices

Monolithic, tightly-coupled applications exist in most organizations because, for a long time, that was the best software design approach we had. As our industry evolved though, we became increasingly aware of the constraints of this architectural approach:

- Build, test, and deploy require large batches of work
- Dependencies need to be managed - both people and systems
- Dependencies cause fragility
- Fragility causes bureaucracy (e.g. **CABs**)
- Bureaucracy and handoffs slow flow

Just like with changing the human behaviors in a system, rearchitecting applications to microservices doesn't happen overnight and there are frequent hiccups along the way. This is commonly a 'change or die' conversation; if the application itself doesn't become more capable

of change, market-disruptors will make new experiences available to consumers and customers will be lost to these competitors.

A value stream management platform provides insights into where constraints, delays and dependencies exist - and can be improved. There is a tenet of DevOps that says, **“Don’t manage dependencies; break them.”**

However, the reality for many organizations is that the road to systemic autonomy, or loosely coupled systems, is a long one with many small increments of improvement. The strangler pattern advises this approach; chipping away at the monolith, one piece of functionality at a time (like a vine).

We need to support value stream teams during the application architecture transition so that they are not spending time on unplanned work caused by unseen dependencies. And we need them to be able to see and celebrate the improvements they make. A value stream management platform provides this support.

As the organization transitions, part of the release manager’s role is to identify value stream success stories and share them across the other value streams so that local discoveries can become global improvements.


3.1.3 Transition the Release Management Role to the Value Stream Team

When teams are working autonomously with small changes they can release them when they are ready. But it takes time to transition to that stage in the journey as you move from one state to another.

In a traditional way of working, there is likely to be a release manager or a team of release managers who are coordinating the release process. The release manager role can be transitioned to the value stream team, with the team giving the release manager access to systems that provide visibility into current releases.

It's always hard to let go of something over which we have traditionally had control. It's hard to trust someone with what we have previously been accountable for. But this is what we are asking leaders throughout businesses to do when we distribute authority and empower value stream autonomy.

Having transparency into the activities across value streams gives a great deal of comfort to traditional release managers during this transition, freeing them from the burden of administration. It gives them the time to find ways in which to evolve themselves to hone the meaning and purpose they find in their own work.



“Culture is a big problem when adopting these ways of working. Release is the last stage before it hits the customer and there’s a sense of fear; ‘choke-con’. **Who hears first when there’s a problem and who has the problem pinned on them?** This results in trust issues. People also need to be aware that if you change only the release processes, the rest of your functions for building, planning and adjusting may lag behind.”

- Mark Peters, Cybersecurity Expert & DevOps Leader

Release managers can be critical to the success of these changes by taking an end-to-end and cross value stream approach to change and improvement and ensuring blame becomes accountability. Value stream management platforms provide the data to support the conversations that make this cultural and behavioral change happen.

3.1.4 Increase Availability of Release Slots and Standardize Change

Initially, when teams start using agile frameworks such as **Scrum**, they will aim to release at the end of a two-week sprint, or perhaps at the end of several sprints. If the organization is working with a release calendar that may have quarterly or monthly release slots, they should look to increase the number of slots available to allow for the smaller and more frequent changes.

In time, whether teams are using **sprint or Kanban** ways of working, they will evolve to releasing on-demand or continuous delivery. At this point, no type of release calendar or management is required as the teams operate autonomously.

At this stage, release managers work within the value streams to create different categories for smaller change increments, e.g., standard change. As teams prove that their changes are well tested and are able to release without drama, release managers can allow them increasing amounts of autonomy.

Coordinating all of this through a value stream management platform means that when things go wrong (which they inevitably will, but less frequently and with reduced impact) in value streams the release manager, or the team (if the role has been fully subsumed), can swiftly identify the cause and remediate.

3.1.5 Automate the Release ‘Checklist’ and Deployment Process

Many organizations operate with a release checklist to ensure agreed policies and procedures are met. Many of the steps throughout transition include automating the release checklist within the CI/CD pipeline, such as versioning and testing, allowing teams to release themselves; they build it, they own it.

Throughout these changes, release and deployment autonomy is highly dependent on system autonomy. Where systems remain tightly coupled, release management tools are available to track and manage these system dependencies.

These systems can also profile the risk associated with release. Deployment automation or orchestration tools as part of the CI/CD pipeline further predictability in the process by reducing the manual effort associated with these tasks and providing patterns or templates that reduce configuration drift and allow for self-service in the teams.

This is also a core capability of a good value stream management platform. Value stream teams should look for capabilities where workflows and access control models allow for this automation.

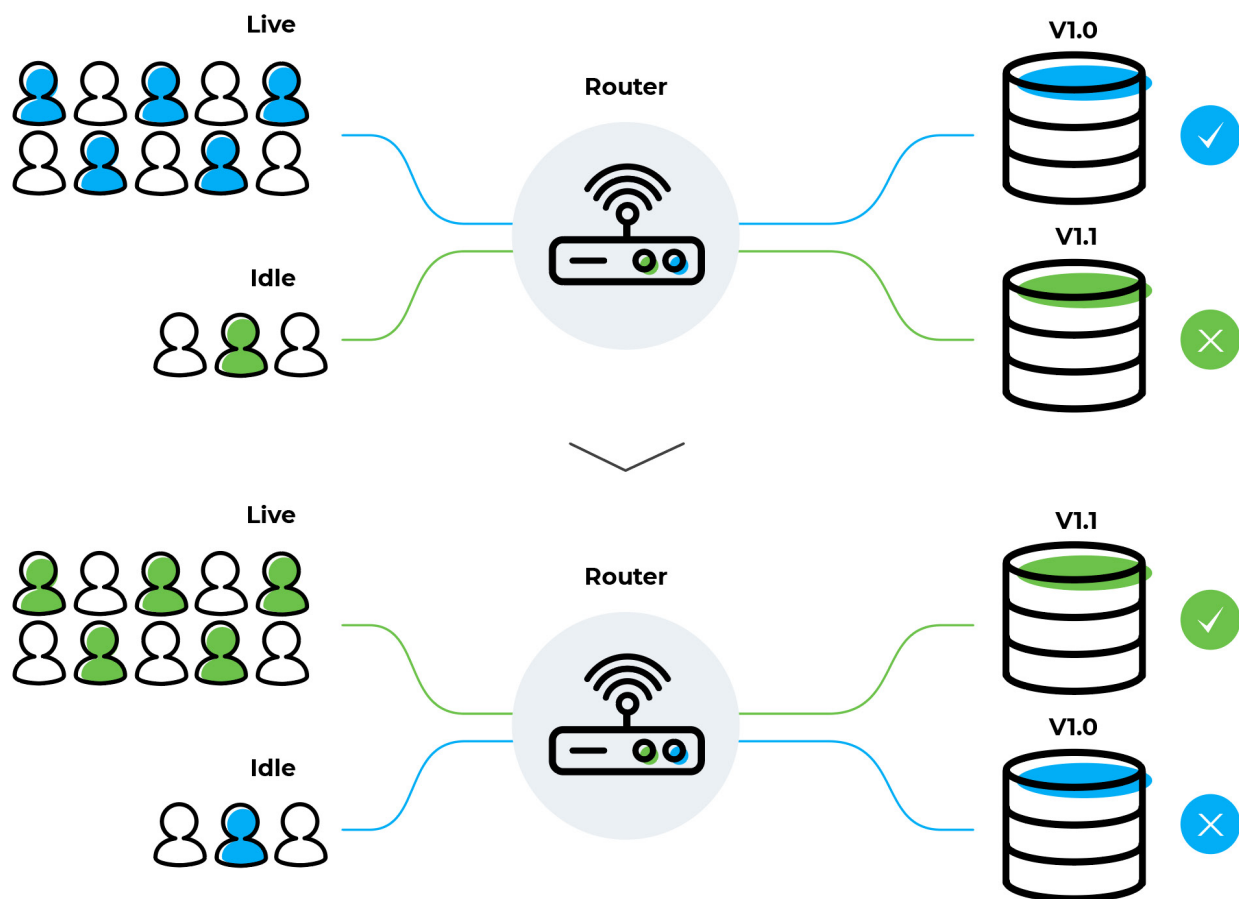
3.1.6 Limit the Deployment Blast Radius: Dark Launches

Organizations use a **canary testing/deployment** scenario and also use feature toggles/flags and blue/green deployments to mitigate deployment failure risk.

Feature toggling separates feature release from code deployment allowing code to be deployed to production while restricting access (through configuration) to a subset of users. In canary testing we deploy to a small number of users and check their experience is great before we proceed with deploying it (in waves) across the entire user base.

Together this allows unfinished code to undergo integration testing whilst remaining inaccessible when live, allowing for A/B testing and canary testing/deployment.

Blue-green deployment is a technique that reduces downtime and risk by running two identical production environments called Blue and Green. At any time, only one of the environments is live, with the live environment serving all production traffic. For this example, Blue is currently live and Green is idle.



As a new version of the software is prepared, deployment and the final stage of testing takes place in the environment that is not live: in this example, Green. Once the software is deployed and fully tested in Green, the router switches incoming requests to Green instead of Blue. Green is now live, and Blue is idle. This can also help with reducing the Route to Live (RtL) which reduces handoffs and opportunities from problems and improves flow.

These techniques greatly reduce unplanned work caused by incidents. Value stream management platforms provide insights into how unplanned work interrupts flow and allows teams to measure how they have improved as they have adopted these techniques. These techniques should be of particular interest to a release manager as they significantly reduce business risk.

3.1.7 Reducing the Route to Live, Leveraging Cloud

Many organizations have complex RtLs containing multiple test environments and experience difficulties in production since these environments are not production-like. Teams also frequently have to share these environments and find it difficult to obtain good test data.

The factor that most commonly prevents teams from having access to production-like test environments is cost. Using **cloud technologies** can ease the pain here (and research shows that using these types of technologies (public, private, hybrid or multi) correlates with higher-

performing organizations) allowing teams to easily spin up test environments when they are needed.

Working in small increments, using blue/green deployments, automating testing, embedding testing in the team and Test Driven Development (TDD) all contribute to a reduction in the number of steps in the RtL. This reduces risk and accelerates the flow of value. Once more, value stream mapping uncovers how much time is spent stepping through the RtL.

TDD is a software development process that relies on the repetition of a very short development cycle: first the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test, and finally refactors the new code to acceptable standards.

Value stream management platforms provide insights directly into what these RtL configurations mean for time to value for the customer. Release managers can take the best RtL patterns and help other value streams to adopt them.

3.1.8 Keep an Eye on People, Behavior and Cultural Challenges

We often define DevOps using the CALMS acronym, where the 'C' represents culture. Research regularly reveals that the biggest challenge organizations have when they are adopting DevOps ways of working is a cultural one. It's very hard for humans to let go of old ways of working, habits and behaviors and invest in the learning needed to change.

"Culture is the biggest barrier to making this sort of change happen. There is fear that something new will not provide the level of service assurance that a people centric management function has proven. Stakeholders are often scared that their kingdoms will shrink, so focus on the productivity/cost savings and the injection of innovation.

Release managers themselves are fearful of change; redundancy, their own sense of self. **But everyone needs to understand there are only two choices: evolve or die.** Not everyone makes the cut; "If you can't change the people, change the people."

- Ryan Sheldrake, Release & Environment Manager

Nadeem Augustine agrees:

“The biggest challenge is always the people and helping them leave behind legacy ways of working. They are used to preparing for a release date three months out. This demands a significant shift in mindset.”

- Nadeem Augustine, DevOps & Release Lead

3.2 A Note on ITIL4

Many IT Operations teams have been working with **ITIL** for many years. Recent releases seek to modernise release management ways of working.

“ITIL4 asks questions like: “What can the consumer bear? What can the infrastructure bear? Is capacity available?”. It asks teams to test for success and focus on value and outcome principles. It has a value stream focus, accounting for all the experiences end-to-end. It drives two-way conversations.”

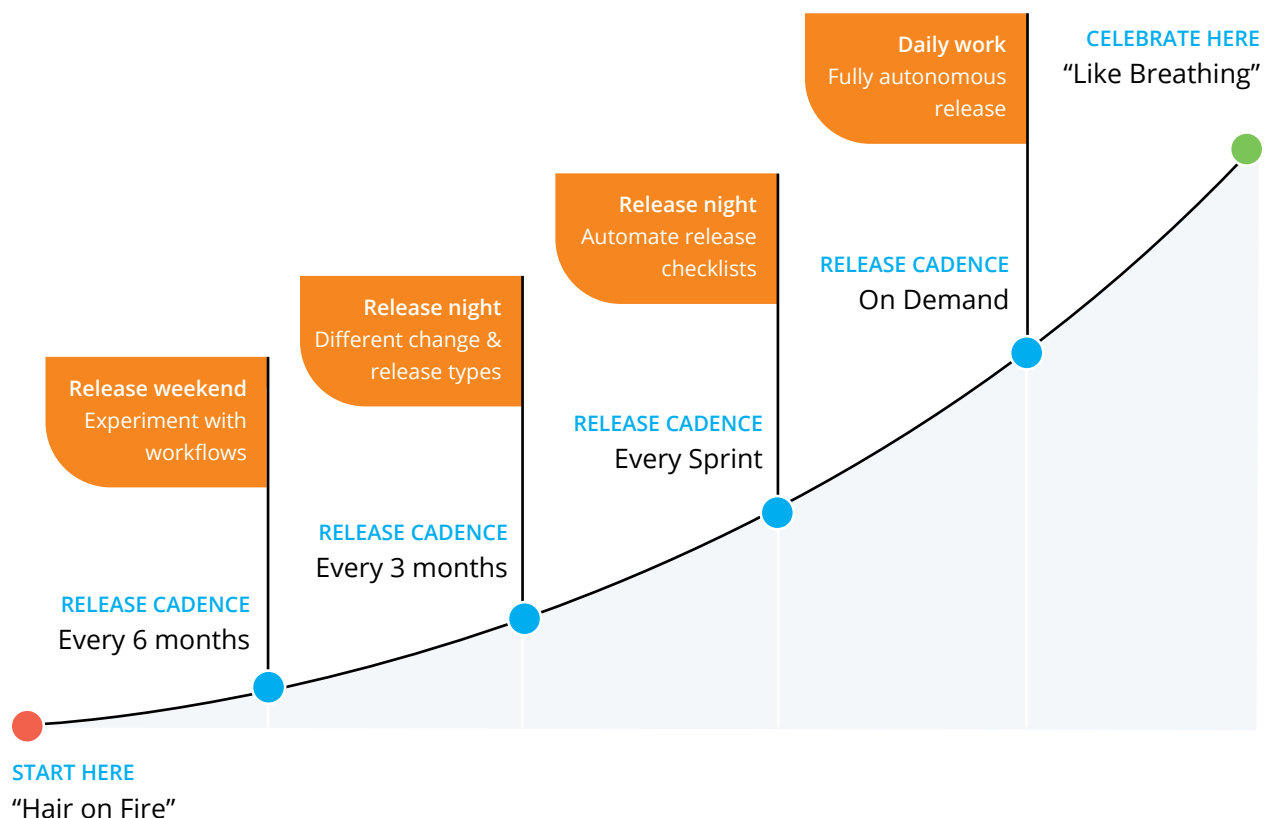
- Simone Jo Moore, Transformation Consultant

ITIL4 is a widely used framework with a reputation for being onerous. This typically happens when teams try to adopt it verbatim without customizing it to their specific needs. In the most recent updates, it has become more value stream centric and focused on agility and velocity.

3.3 DevOps Release Management Best Practices

Existing release managers may feel threatened by the change happening around them, but here are some best practices to thrive during your organization's transition:

- **Be a leader of the change:** your role is pivotal
- **Define and articulate your vision** for release management
- **Figure out how the value stream(s) exist around you** - even if your organization has not yet defined its value streams
- **Use a value stream management platform** to ease your administrative burden and fuel data driven conversations around improvements
- Aim to achieve **continuous compliance**




3.3.1 Connecting Planning

Value streams run from idea to realization and release managers need to incorporate planning into their way of working to gain visibility into upcoming releases. Release managers need to be able to understand when releases are coming and their implications. Throughout your DevOps journey it is critical that release managers communicate the business risk, customer value and change impacts of upcoming releases by collaborating with planning teams, for example in SAFe's PI planning.

3.3.2 Data Driven Conversations

Throughout the transition to DevOps release management, it is important to motivate change through data-driven conversations. These conversations reduce blame and increase accountability for improvement experience. Tony Mongiovi, Release Manager at HealthFirst said:



“Prior to Plutora [a value stream management platform], people frequently asked for more time for functional testing and they would be fire fighting. But now, I can see which pieces haven't been accepted into the release by the respective product owners.

I can then have a data driven conversation with them and find out what the problem is, allow extra time if needed or defer to a future release. Then we can go back, inspect what happened and gain learnings and insights for improvements. We still have governance and we still have oversight - but it doesn't get in the way."

- Tony Mongiovi, Release Manager at Healthfirst



3.2 Use Cases for the Release Manager

“As a Release Manager I want to be able to easily see all of the release trains that are running so that I can manage dependencies.”

“As a Release Manager I want to be able to see and understand the dependencies between value streams so I can support the teams in breaking them.”

“As a Release Manager I want to be able to easily report on all the status of features in all of the tools in the CI/CD pipeline or DevOps toolchain so that I can manage risk.”

“As a Release Manager I want to have access to the data that shows me the teams are acting within the guardrails we have agreed and performing the checks we know are necessary to operate safely.”

“As a Release Manager, I too care about the speed and quality of the value the customer is receiving. I want to be able to support teams by highlighting potential problems before they happen.”

3.2.1 Example Experiments

As the team transitions to new ways of working they should include experiments as part of their improvement kata as you determine the next target state. Examples of these experiments include:

“We believe that if we set up blue/green deployments for our product then our change fail rate will drop by 2% within 6 months.”

“I, the tester, think that if I work with the developers on my team to have them all practicing TDD by the end of this year, next year our rework traced to defects will reduce by 10%.”

“Our product team will implement release and deployment automation over the next 3 months. We anticipate that once this is in place our cycle time will improve from 33 days to 10 days.”

“I believe that if the team can show our release manager that we have passed all the checks required in our release policy and she can see our planned releases and performance, we won’t need her to give us permission.”

“We think that implementing a deployment automation solution will reduce our change fail rate by 5% and reduce the time to release from 28 hours to less than one hour. This means our deployment frequency will quadruple next year.”



4.0 Release Management Automation

A lot has been written about DevOps automation. We are seeking to automate every possible step in the pipeline (manual steps that are repeated or toil) and typically this results in building a DevOps toolchain consisting of many parts in these typical tool categories:

- Planning & portfolio management
- Requirements and backlog management
- Artifact repository, version/source control
- Build/CI server
- Unit testing
- Integration testing
- User acceptance testing
- Security testing
- Environment orchestration
- Release & deployment orchestration
- Service desk
- Observability, monitoring, logging, alerting & AIOps

Whilst this is a list, it should be thought of as a circle, where the feedback from the customer feeds back into the requirements.

“Release management in DevOps is all about deployment quality. It’s about quicker coordination, being faster to production. Release train engineers bring the pieces together. They make sure they have repeatability. That’s the ability to make sure that not only deployments but the processes are repeatable to ensure a quality way of working - reliable and repeatable - creating a good working model.”

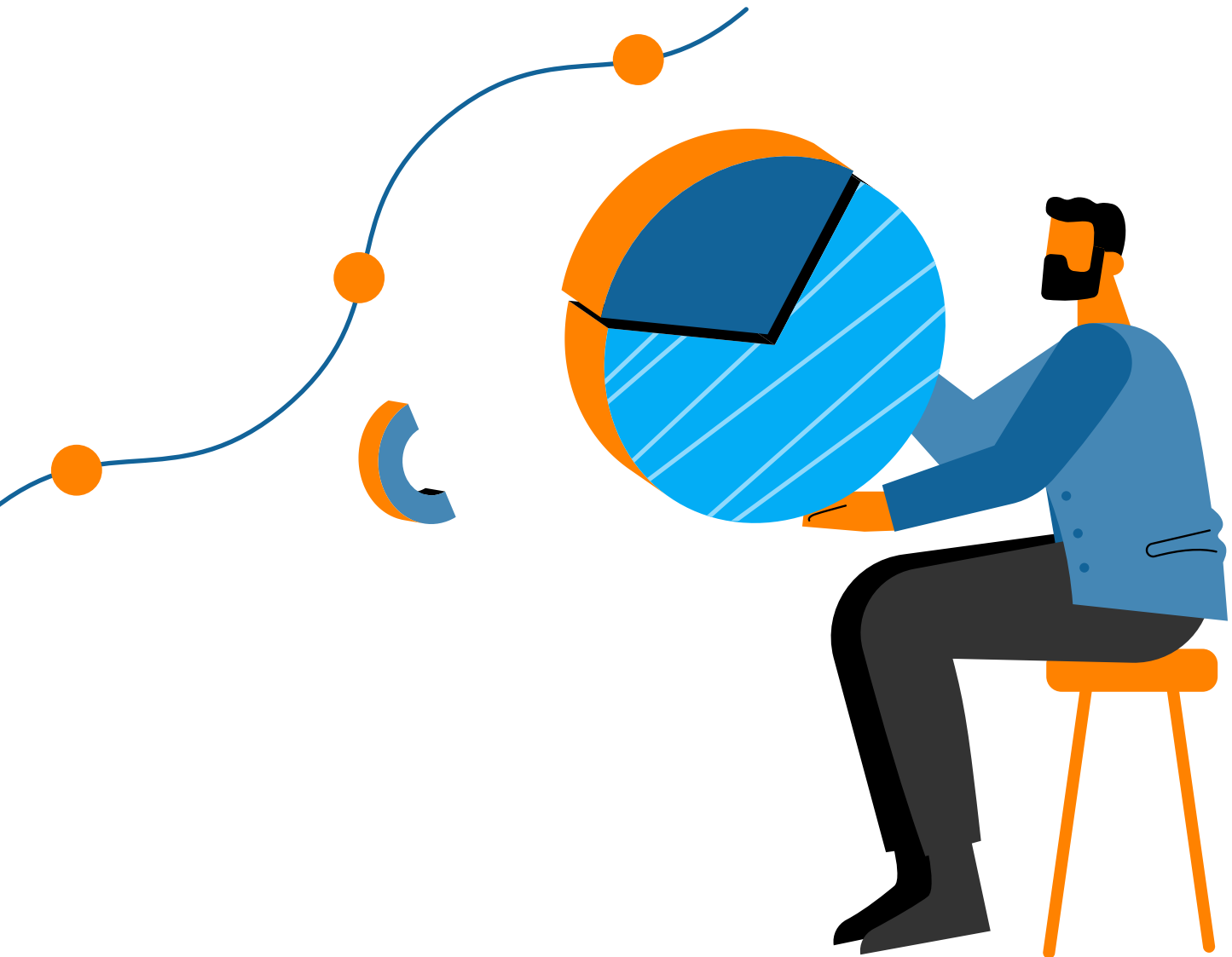
- Nadeem Augustine, DevOps & Release Lead

Release orchestration is included with deployment orchestration, but there is so much a release manager needs to consider that can be automated outside of the action of making software live:

- Coordinating multiple teams
- Making autonomous teams visible
- Having transparency across multiple value streams
- Managing dependencies in tightly coupled environments

Value stream management adds a systems-thinking approach to DevOps release management. It aims to unify the CI/CD/DevOps toolchain with a value stream management solution to enable a holistic perspective in release management through:

- Traceability and integration
- Data, telemetry and observability
- Data-driven journey mapping



5.0 The DevOps Release Manager

The new, DevOps oriented, release manager is a different breed from what has come before. To evolve their ways of working, every release manager should look beyond their day-to-day work to what they can do to assist their organization in achieving their vision for thriving through digital transformation and adopting DevOps and value-stream centric ways of working.

“The release manager used to be the gatekeeper; the role is shifting from ‘bouncer’ to ‘maitre d’.

Release should now be delivered by pushing a button; going live is still critical but not such a big and dangerous event.

There are no more checklists since they are automated within the process. CABs take a different form as there is a different team and organization structure. The focus is on enabling the change with the right people engaged in the conversation. Effort and energy is invested where it’s needed. Requirements are built together, early.”

- Simone Jo Moore

“The role shifts left. It moves from being about paper-pushing to starting to write tests and being a business risk manager. In the new world, release managers need to be technically astute. That means being engineering aware or capable - not necessarily a coder, but being able to understand, verify and work with a test. In a multifunctional team, theoretically the release manager is everyone. In transition, someone still needs to be accountable.”

- Ryan Sheldrake, Release & Environment Manager

The Old ITSM Release Manager	The New DevOps Release Manager
Gatekeeper, bouncer, manager	Enabler, maitre'd, coach, overseer, guide
Large and dangerous release events	Small and safe releases
Runbooks and checklists	Automated, continuous compliance
CABs	Peer-reviewed change
Calendar driven	Conversation driven
Focus on project, cost, budget	Focus on product, value, flow
Administrator, paper/process-based	Engineer, automator, technical skills
Separate team, role, individual	Integrated individual, role, everyone
Manual reporting via multiple sources	Automated feedback through VSM
Schedules, wait times	Invisible, frictionless


5.1 Tips for Release Managers Wanting to Evolve

Traditional release management roles are likely to be around for a while yet, but they are evolving and may disappear entirely.

This requires release managers to take the initiative to collaborate with the business, communicate release impacts, and continuously improve through data-driven decisions.

There are a lot of different metrics that release managers need to gather to drive improvement, and that they come from a lot of different categories in the DevOps toolchain (CI/CD pipeline, APM/AIOps, ITSM, other parts of the DevOps toolchain). Using a value stream management platform saves release managers a huge amount of time in terms of data collection and expedites insights for improvements.

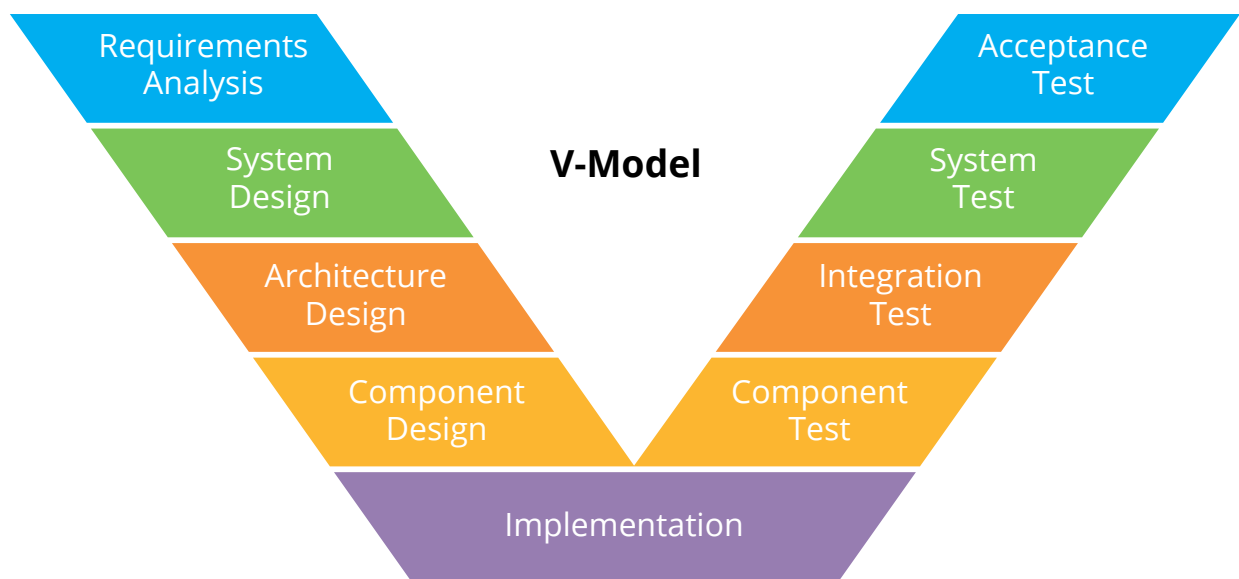
Moore said:



“It’s all about creating transparency. The V-Model is still useful in testing at every level but a must have is early testing; testing at the point of creation. Have the value stream workshop to ensure that everyone understands the activities that need to take place,

who is involved, what they need to do, how they are integrated and need to collaborate and how they will improve. Make sure that everything fulfills the Why (long term vision and purpose). These things are not 'carved in stone' but adaptable - dynamic, living, breathing, iterative processes."

- Simone Jo Moore



Understanding the value stream is certainly critical, but for the longevity of this way of thinking and working, look to value stream management platforms to automate the value stream map. They also provide the environment for the value stream teams, including release manager roles, to inspect, adapt, and measure their progress on their DevOps journey.

“Identify where your personal interests lie. Discover ways to broaden your skill set. Ensure you can articulate how you deliver value. Position yourself as an engineer. Eradicate the silent business killer that is waiting! **Release management is then easy - or doesn't exist.**

Visibility - find ways to monitor the system (rather than have a system of record about the system)
- ‘agents’ that collected the system data and aggregated into the environment management portal - autocheck across multiple systems really quickly and identify issues.”

- Ryan Sheldrake, Release & Environment Manager

In many organizations' vision, the release manager role no longer exists. Teams can release autonomously (they are multi-functional, the systems are loosely coupled. Risk has been automated to minimum levels with continuous compliance. Release management, which was never a value-adding activity, has become redundant and the time and energy once spent on this administrative activity, is now available for innovations to delight customers.

Peters shares a view with many on where DevOps will take organizations and release managers:

“Why are we using a release manager? If you have specific release managers, you have already left DevOps behind. A release manager is a constraint on your ability to flow the process. With transparent standards and observability, anyone in the organization should be able to release. Where do they go? Hopefully they'll have the skills to work back into the teams; maybe a scrummaster, a developer, a SRE.

You don't transition right away; release managers start out by explaining the approach to teams, then training them and finally hands-off their process off to the teams. Eventually you're going to automate a significant amount of that release so you don't have someone sitting in it. In a traditional environment it is automated, but only one person has the key. In DevOps we make copies of the keys - just like with environment provisioning. You have to have enough trust.”

- Mark Peters, Cybersecurity Expert & DevOps Leader

5.2 New Roles for Release Managers

Change is constant in life and nobody can afford to not look after themselves. The world changes around and it's up to every individual to choose how to keep up. Whilst many traditional roles are receding, even disappearing as organizations strive to adopt these new ways of working, people should see it as an opportunity rather than a threat.

There has never been a better time to upskill; technologies are more accessible than ever. Both teams and individuals are becoming increasingly multi-functional and multiskilled and there are opportunities for all to find and pursue new purpose and meaning in their work. Here are some options for Release Managers looking for a new role as theirs recedes:

5.2.1 Site Reliability Engineer (SRE)

Site reliability engineering predates DevOps from when it was incepted at Google - but it wasn't released publicly until sometime after the DevOps movement had taken hold. Most organizations then are starting to understand what SRE means for them and closely connecting their investments to their DevOps journey.

Release managers who are interested in automating toil and focusing on risk and reliability should consider evolving to SRE.

5.2.2 Release Train Engineer (RTE)

As agile adoption increased in organizations, so did the realization that early frameworks lacked the ability to scale. A number of agile at scale frameworks have appeared in recent years, some of which use the concept of release trains to allow multiple value streams, with dependencies, to coordinate activities.

DevOps aims to break dependencies, not manage them, but the realities of moving between ways of working during a DevOps journey means that this is a useful mechanism during transition. It's not without its challenges though, as Augustine said:

“Our processes need to adapt to the way we deploy in a modern day world. Waiting period, governance, release gates - these are now automated into the CICD pipeline. Because the teams are disjointed, it's hard for the release train engineers to perform the deployments.”

- Nadeem Augustine, DevOps & Release Lead

Release managers who enjoy the complexity of system dependencies and ultimately want to work with architects to loosely couple applications and have or want to acquire lower levels of automation engineering skills should consider **release train engineer** roles.

5.2.3 Product Owner

Release managers who enjoy collaborating with the business and particularly enjoy seeing new innovation become available for customers to experience should consider a move into a product owner role. Being accountable for refining the backlog and prioritizing with the stakeholders and team will come naturally to many release managers familiar with juggling competing resources and the need for clarity.

5.2.4 Value Stream Architect

A value stream architect's primary concern is increasing the flow of value to the customer, whilst not increasing business risk. This is the bread and butter of a release manager and they are likely to find the end-to-end nature of this role compelling. It will give them insights to part of the process they likely have never seen before and hugely rewarding opportunities to make a difference to their colleagues and customers.



5.3 The Future of Release Management

Perhaps it's too much to look to the future when the future for so many may be the states already described in this paper. That's why we recommend talking about capability, not maturity.

This is partly because different value streams will have different capability profiles but also because, in this industry which is so fast-moving, with innovation happening all the time, we never get a chance to get old; only left behind.

Our horizon constantly moves as we travel towards it. Here are some examples of advanced capabilities that most organizations have not fully adopted as yet, that give us some ideas about where the future may be taking us:

5.3.1 Value Stream Management

Value stream management is an emerging practice and platform, but in 2020 we saw a number of analysts call it out as essential to the adoption of advanced DevOps practices. It also provides teams and organizations shortcuts to capabilities that should make investment in the practices and platforms very appealing:

- Visibility into cross value stream dependencies
- Automated value stream maps for inspection and adaptation
- Traceability through the end-to-end DevOps toolchain
- Insights for data-driven conversations and automated audits
- A framework for **continuous compliance**

5.3.2 AI and ML

Many organizations are in their infancy when it comes to adopting artificial intelligence (AI) and machine learning (ML). But these technologies have so many use cases already for DevOps oriented value streams:

- Autosuggestion of code snippets for developers
- Improving requirements accuracy using NLP
- Bug detection and autosuggestion for improvements for testers
- Vulnerability detection and triage
- Autogenerating and running test cases

All of those use cases should be of interest to release managers as they reduce the business risk associated with release, but there are release management specific ways in which AI is already helping.


Value streams can use AI to understand why some releases go smoothly while others find themselves in perpetual review and code rewrite cycles. Supervised machine learning algorithms identify patterns and insights into the data.

AI can help value streams to coordinate with each other better, particularly when geographically dispersed. AI-driven insights provide visibility into how shared requirements and specifications can reflect localization, unique customer requirements, and specific performance benchmarks.

Sheldrake said:

“We know about “fail fast, fail safe”, it’s been around for ages. I think AI/ML (machine learning) are starting to factor into the release processes. Imagine a “standard” release e.g. a new version of a webpage. The machine can baseline the deployment, performance, security and regression testing etc. building up knowledge automatically over multiple releases of the same taxonomy.

Now imagine that the baselines that cover this complex release enable a machine to re-order the



engineering for analysis and re-queue by committing an amendment(s) to auto deploy upon next commit/iteration.

This brings big cultural changes and engineering practice amendments. I mentioned “RDD”(Release Driven Development) but this goes down to a lower level. Think about backout first, then test, then develop the feature.”

- Ryan Sheldrake, Release & Environment Manager

Further Reading

Learn how to use value streams to navigate and accelerate the DevOps transformation of your software delivery factory, whether it be on-site, remote, or any combination of the two.

Become a software juggernaut.

[What is Value Stream Management?](#)

[Learn DevOps: Enterprise DevOps at Scale](#)

[The Secret to DevOps Success: Release Management](#)

[CI/CD Tools Universe: The Ultimate Guide](#)

Discover the power of value stream management with Plutora:

[Why Plutora?](#)

[The Plutora Platform](#)

[Request a Demo](#)