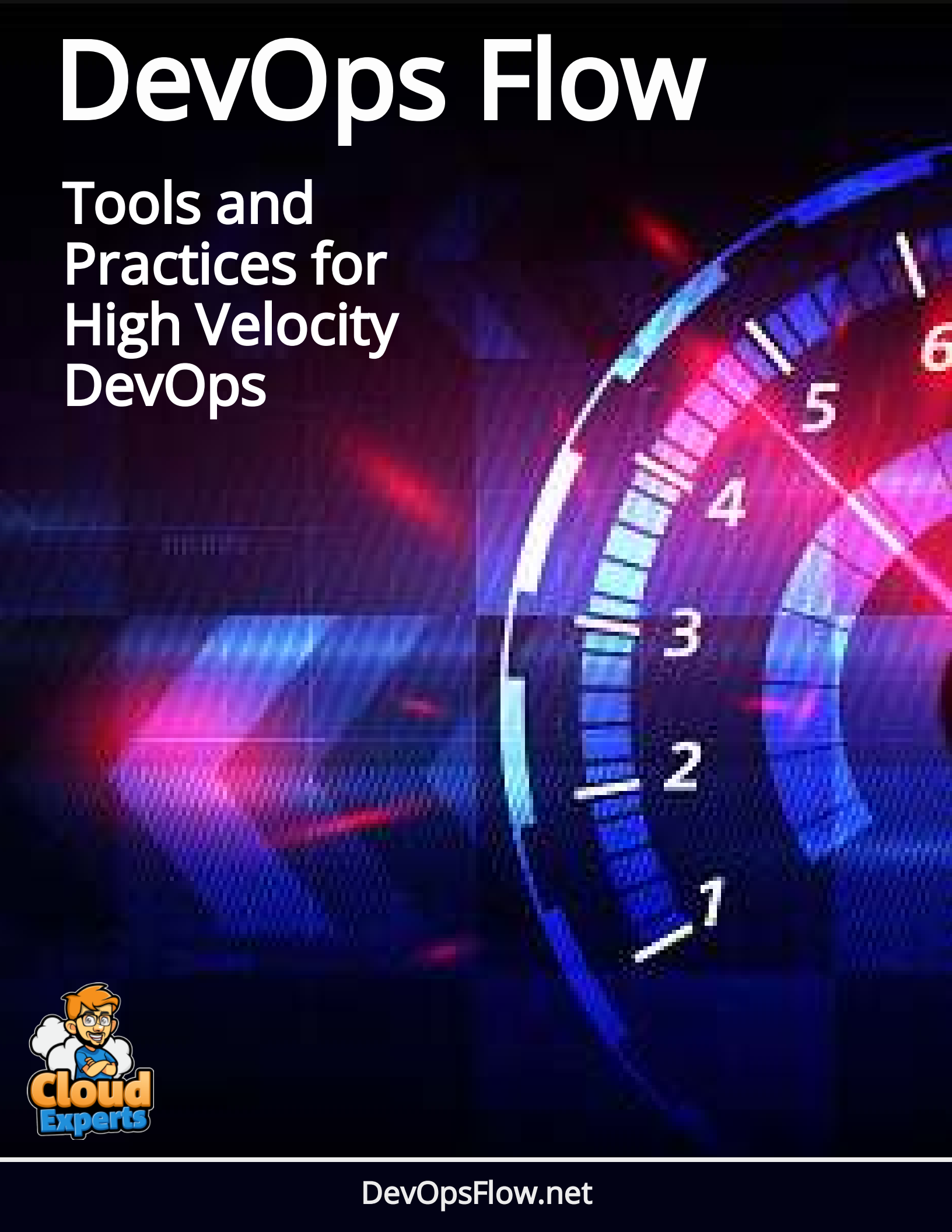
DevOps Flow

Tools and Practices for High Velocity DevOps



Cloud Experts

DevOpsFlow.net

DevOps Flow - Accelerating Developer Velocity Through Software Factory Best Practices

In the Cloud era senior executives are investing into key capabilities like DevOps with the expectation this will accelerate innovation and bring new digital products to market faster.

To deliver on these goals enterprise organizations seek to deploy high quality code, faster and more frequently, but they face a complexity of organization, process and technology that can hinder this objective, slowing throughput and incurring high error rates.

Often when DevOps is discussed the focus is concentrated on the technology layer, particularly the latest tools like Kubernetes, and also when considering performance it zooms in on the productivity of each individual developer.

However for many organisations the main challenges will be at the level of their team practices and organizational models.

Writing for DevOps.com Mike Vizard highlights the reality of DevOps uptake, that most adoption issues are related to organisational challenges, not technology. The top two barriers to adoption are slow processes and speed of adaptation (29%), followed by budget and funding (21%). Only 18% identified technology limitations as an issue.

DevOps Flow - Accelerating Developer Velocity Through Software Factory Best Practices

As Mike writes it is the broader organisational transformation that is more challenging. 'Addressing bottlenecks' might be a function of confronting the situation of a department manager who zealously guards their turf and acts politically to resist any changes required.

Implementing DevOps for one team is a relatively straight forward exercise, but for enterprise organisations, they face a much larger magnitude of complexity, as they typically have multiple teams, spanning multiple geographies and suppliers, with hundreds of developers all contributing to the same software development process.

Therefore a DevOps methodology that also addresses the organizational level is required, one that maps processes end-to-end and identifies the improvements needed to speed production across this entire life-cycle.

Software Factories

Transformation leaders are turning to a '**Digital Factory**' approach to streamline this complexity into a production line model that yields consistently high throughput and quality, using Lean principles derived from the world of manufacturing.

In their in-depth article, McKinsey explores the dynamics and successes of the Digital Factory model, a new approach to organising digital teams, a self-organising structure vs rigid departmental hierarchies, and they describe the KPI improvements the Digital Factory enables:

DevOps Flow - Accelerating Developer Velocity Through Software Factory Best Practices

> *"We see reductions in management overhead of 50 percent for technology teams in the DF, 70 percent in the number of business analysts needed to write technology requirements, and, as test automation becomes the norm, a drop of 90 percent in the number of testers. Finally, we see top engineering talent performing at eight times the level of their peers, as measured with metrics such as code commits."*

In short McKinsey describe an improving development team that improves quality and finds efficiencies is a direct by-product of one that leverages the latest tools and team methods to produce better digital products, faster.

The concept of a 'software factory' is a very powerful metaphor because much of the insights are drawn literally from the world of manufacturing, where over many decades they have perfected the science of optimizing the efficiency of a production line.

To close the loop and map these development improvements to Business Value, management can leverage organisational performance practices notably Six Sigma and the Theory of Constraints to define their 'DevOps Algorithms', a systematic formula for quantifying and improving the rate of software development throughput.

DevOps Flow is a methodology for implementing high performance, Cloud-centric software development, based on a science developed by pioneers like Toyoto to optimize factory lines and apply this to software engineering.

DevOps Flow - Accelerating Developer Velocity Through Software Factory Best Practices

Organisations need the ability to measure the entire system end-to-end to understand how value flows and where it is constrained, and most importantly, to correlate those metrics with desired business outcomes. This approach allows for continuous optimization in the pursuit of delivering greater and greater value to the organisation, faster.

It can be applied to speed the deployment of new software releases, notably identifying and removing constraints, reducing batch sizes and eliminating waste. These are all steps taken within manufacturing to increase production throughput and are improvements that software teams can adopt to achieve equivalent benefits.

Establishing a DevOps Centre of Excellence

DevOps Flow is best implemented through establishing a DevOps Centre of Excellence, a central structure that can facilitate the adoption of industry best practices that drive efficiency and the embracing of Cloud practices.

This can act as a catalyst for training, knowledge transfer and cultural shift assistance for your teams and accelerate faster business innovation and reduced time-to-market through high-performance DevOps teams.

Transformation Roadmap

The first step is a maturity assessment to map where your organisation currently stands and from that plan a roadmap for moving forward.

DevOps.com offers an article that explores the types of challenges the process will face, made up of three main elements: organisation structure, legacy technology stack and culture.

They observe a common pattern many organizations will begin with, that typically there are several small DevOps programs already in place in silos that lack the maturity to scale to the enterprise level. Therefore one of the first primary steps and critical success factors is that it is championed at the senior executive level and be chartered as an enterprise-wide program.

Following on the program governance should reflect this ambition, this is the most important step to make an impact on the culture of the organisation. Roles and responsibilities will change and this should be guided by a holistic understanding of the collaboration needed across different business units. KPIs must shift from individual metrics to holistic customer business outcomes.

For an overall set up process InformationWeek defines a four-step approach:

1. **Assemble a team** – Build the required mix of team, who can define the core body of knowledge, engage stakeholders and train new recruits.
2. **Create a knowledge and learning hub** – The DevOps Center of Excellence should define the organisation's best practices and teach these policies and procedures to all practitioners.
3. **Define the toolset** – Define what the standard DevOps toolset is across the organisation and monitor teams to ensure they are adhering to that standard.
4. **Let automation drive DevOps** – The DevOps Center of Excellence should always be exploring how to further automation can improve DevOps procedures. The team needs to stay abreast of the latest advancements in the space and continually ask how they can make the process more efficient, more secure and more reliable through automation.

The Enterprisers also offers four points, that take an action-oriented implementation perspective. For example, the first task is to choose the right use cases, identifying where it will have the most impact, and through the next three steps continue this emphasis on the importance of a business-aligned COE, with clearly defined business outcomes, such as increased flexibility or cost savings.

DevOps Metrics - Defining Measurements for Achieving High Performance Software Development

The principle function of a DevOps Centre of Excellence and the critical framework for defining and achieving high performance are process metrics.

Establishing the right ones will enable development teams to define and measure what matters, providing a control loop for ensuring their transformation and new working processes deliver the results they want.

To determine the business case for Continuous Deployment practices the key dynamic is 'Business Value Throughput'. Ie. not just speeding up the production of deployed code but of software that adds quantifiable value to the organisation.

In his DevOps.com blog, Matt Dickens also explores the same area, providing a very helpful guide to 'Choosing the Right Metrics for DevOps Adoption'.

Matt starts with the key point that the transformation is an ongoing, long term process of transformation, and like always the success factor is having a clear end goal in mind and then breaking it down into a series of 'bite size chunk' steps to progress along that journey.

His article is very helpful because he provides an initial list for suggestions of what these steps might be for different organisations at different stages of their adoption, and then maps each step to the most effective tools and practices improvements to make to progress against that specific metric, such as:

- Deployment time
- Change failure rate
- Release cadence
- Lead time
- Recovery time

Flow Metrics

In the feature video Mik Kersten, Founder and CEO of Tasktop, describes his journey and evolution as a software developer that led him to understand what is required to achieve high performance software development teams.

He has encoded that learning into the principle of Value Stream Networks that can be implemented and managed through Flow Metrics.

There are four Flow Metrics that measure how value flows through a product's value stream, calculated on four Flow Items – units of work that matter to a business: features, defects, debt, and risk. Any task or effort a software delivery organization undertakes can be categorized as one of these core Flow Items.

- **Flow Velocity** gauges whether value delivery is accelerating also referred to as throughput.
- **Flow Time** measures the time it takes for Flow Items to go from 'work start' to 'work complete', including both active and wait times.
- **Flow Efficiency** is the ratio of active time vs. wait time out of the total Flow Time and
- **Flow Load** monitors the number of Flow Items currently in progress (active or waiting) within a particular value stream.

DevOps Metrics - Defining Measurements for Achieving High Performance Software Development

Tasktop describes the central challenge and the need for a new, better approach for capturing and reporting on productivity:

"While IT frequently collects and presents an abundance of technical metrics regularly, quite often they measure the process and not the outcome. Also referred to as 'proxy metrics', these metrics include things like story-points delivered, commits per project, test case coverage, build success rate, release duration, and deployments per day.

Good measures of a process are helpful to monitor the performance of a specific activity, but they usually don't help IT and business leaders optimise the system's performance as a whole, from customer request to delivery. Furthermore, optimising something that is not the system's bottleneck is actually inefficient and counter-productive."

Project to Product

Mik has also documented his insights into a book Project to Product, and in this talk he summarizes it by showing how a product operating model can provide the critical glue between the hierarchical and finance-oriented structures of the business and the agility enabled by delivery teams adopting SAFe.

As the title suggests this is centred around a shift from a waterfall, project budgeting approach to one of 'Lean Budgets'.

Lean and value stream thinking originated in manufacturing but have become highly popularized in software delivery by the DevOps movement: With developers releasing code changes more frequently thanks to Agile, DevOps set its sights on getting those code changes running in production faster.

DevOps Metrics - Defining Measurements for Achieving High Performance Software Development

Flow Metrics: Flow time, Flow Velocity, Flow Efficiency

In this webinar Dominica DeGrandis explains Flow Metrics in detail and how to implement these key metrics at an organization and how to enhance the impact of software products.

From 2.08 she sets the scene defining that that flow metrics are tied to business value and are based on outcomes. Flow metrics provide a feedback loop to improve decisions.

At 4:36 Dominica explains the metric, '**Flow Time**'. Flow Time is the duration from when work enters the value stream to its completion. This metric is helpful in answering questions like 'What's the probability of completing work in X days?

This metric helps in identifying when the time to value is getting longer, and is the measure of time taken for the items to process from 'start' to 'complete' state. This would include both active and wait times.

At 16:32 she moves on to discussing the next important metric, which is the '**Flow Velocity**'.

Flow Velocity can be defined as the number of items completed during a given duration of time. This metric is easy to calculate and provides data to the software development teams to view the delivery rates, gauging if the value delivery is progressing in an accelerated manner. This metric is commonly referred to as **throughput**.

DevOps Metrics - Defining Measurements for Achieving High Performance Software Development

At 19:12 Dominica states that there are two important factors that need to be considered while choosing the batch size. Batch size is often decided based on the transaction cost and holding cost. Reducing the batch size of work helps in reducing the WIP (Work in progress) limit and improves the flow.

The optimal batch size primarily depends on holding cost including the cost for delayed feedback, inventory decay, and delayed value delivery). Software development teams must always focus on reducing the transaction costs of any batch.

At 20:51 Dominica explains the '**Flow Efficiency**' metric, the percentage of time where work is in an active state and is a metric to expose wait time. Flow efficiency metrics can easily help in identifying when waste is decreasing or increasing in a process. In other work, these metrics can be defined as the ratio of active time vs wait time of the overall flow time.

At 30:22 she defines the metric '**Flow Load**', a metric to measure the balance between demand and capacity. Flow Load is responsible for monitoring the over-utilized and under-utilized value streams, where often the over-utilization and the under-utilization of value streams can lead to reduced productivity. This metric measures the number of Flow Items that are currently in the active or waiting state within a specific value stream.

At 31:32 she emphasizes that WIP (Work in progress) is a leading indicator, for example a high WIP means that the other items are waiting in the queue for a longer time. The important factor which affects queue size is capacity utilization, the metric to calculate the rate at which the outputs are being achieved.

Queueing Theory

From 33:00 Dominica discusses the 'Queueing Theory', which enables quantifying the relationship between wait times and capacity utilization. As the capacity utilization approaches 100%, wait time increases exponentially.

Software development teams need to consider managing work by queues if the desired goal is speed. At 35:47, she mentions that the amount of WIP (Work in progress) is the primary factor of speed. If there is a requirement to achieve maximum business value, the proportion of work items in a value stream needs to be adjusted. Often a decision to do one thing is a decision to delay something else.

At 43:20 she illustrates a sample value stream dashboard displaying all the flow metrics. The dashboard would comprise prominent flow metrics like Flow Efficiency, Flow Time, Flow Load, Flow Velocity, and Flow Distribution. The dashboards can also have the business value, business cost, business happiness, and business quality index.

At 46:35 Dominica talks about the metric to gauge happiness, a metric that answers questions like ‘How likely are you to recommend working for this company to a friend?

If you are interested in visualizing and optimizing the business value of your software delivery, measuring business outcomes is a must, and Flow Metrics provide a framework for achieving this.

State of DevOps Report - Metrics and Capabilities for Elite Performance DevOps

Google publishes the Accelerate State of DevOps report (2022 version).

Over the past seven years, more than 32,000 professionals worldwide have taken part, making it the largest and longest-running research of its kind.

The report identified four metrics to classify teams as elite, high, medium or low performers based on their software delivery:

- Deployment frequency.
- Lead time for changes.
- Mean-time-to-restore.
- Change fail rate.

They identify four key metrics of high performance delivery, which can be considered in terms of two core objectives: Throughput and stability – How fast reliable code can be developed.

These are measured through i) lead time of code changes (time from code commit to release in production) and deployment frequency, and ii) time to restore a service after an incident and change failure rate.

Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

Elite performers have control over their environment such that only 0-15% of their new changes cause failures, versus 16-30% for the others.

They also deploy new releases much more frequently and can do so in less than an hour. The metrics we described above provide the measurement and management framework needed to improve along with these scales and mature from Low to Elite performance levels.

High Performance Practices

They identify the following technical practices as key to achieving this high performance DevOps:

- **Loosely coupled architecture –** Teams that can scale, fail, test and deploy independently of one another. They work at their own pace and in smaller batches, accrue less technical debt and recover faster from failure.
- **Continuous integration and testing –** Elite performers are 3.7 times more likely to leverage continuous testing. By incorporating early and frequent testing throughout the delivery process, with testers working alongside developers throughout, teams can iterate and make changes to their product, service, or application more quickly.
- **Trunk-based development –** Developers work in small batches and merge their work into a shared trunk frequently. Elite performers who meet their reliability targets are 2.3 times more likely to use trunk-based development.

- **Deployment automation –** When you move software from testing to production in an automated way, you decrease lead time by enabling faster and more efficient deployments. You also reduce the likelihood of deployment errors, which are more common in manual deployments.
- **Database change management –** Tracking changes through version control is a crucial part of writing and maintaining code, and for managing databases. Elite performers who meet their reliability targets
  are 3.4 times more likely to exercise database change management compared to their low-performing counterparts.
- **Monitoring and observability –** Elite performers who successfully meet their reliability targets are 4.1 times more likely to have solutions that incorporate observability into overall system health. Observability practices give your teams a better understanding of your systems, which decreases the time it takes to identify and troubleshoot issues

Practical Challenges

Writing for DevOps.com Mike Vizard highlights the reality of DevOps uptake, that most adoption issues are related to organisational challenges, not technology. The top two barriers to adoption are slow processes and speed of adaptation (29%), followed by budget and funding (21%). Only 18% identified technology limitations as an issue.

Adoption of continuous practices ranges across the scale, with almost half saying they have been able to implement continuous integration, but it then drops away as they assess continuous delivery and then deployment maturity. Only a third of respondents said they deploy new code at least once per week, and almost half still deploy less than once per month. The number of teams that deploy new code daily or multiple times a day (15%) is roughly equivalent to those that deploy code on a quarterly basis (16%).

The most widely used DevOps tools are GitHub, Jenkins, GitLab, Azure Pipelines and Bitbucket. The most widely automated tests are user interface (UI)/functional and unit tests, followed by regression and integration testing, using tools including Selenium, Cypress, Postman, SoapUI and homegrown tools.

Using Tasktop and Flow Metrics to Drive an Integrated DevOps Toolchain

In this presentation Nicole Bryan of Tasktop and Jeff Zahorchak of Select Medical explain the process of mapping DevOps workflow to build an integrated toolchain and optimize it for high performance through implementing Flow Metrics.

Jeff explains that when they started their journey to look at Flow Metrics, they had six main problems they were trying to solve:

- i) “Swivel chair” entering of data into multiple systems. ii) Combined with multiple methods of fielding work requests from a plethora of tools such as instant messages, email etc., forcing them to context switch between many different applications.
- iii) 60% of what the team delivers were incident resolution, so iv) they weren’t working on value adding new features.
- v) Ultimately this meant they were struggling to co-ordinate work across multiple teams, and vi) this required the team to always be over-working to be IT heroes all the time.

Using Tasktop and Flow Metrics to Drive an Integrated DevOps Toolchain

Transformation Journey

Jeff set out to address these issues through a transformation that would:

- Get control of demand through intake management.
- Perform Value Stream Mapping.
- Simplify and integrate delivery tools.
- Implement Flow Metrics to monitor delivery.

The first step was to consolidate all the many different sources of work requests, through developing a custom portal for business users and product owners, with an ROI and approval framework, reducing the interaction methods to three main models: i) An Idea, ii) a Service Request, or iii) an Incident.

From 3m: 45s Jeff explains the heart of the challenge and how they tackled it. He describes a situation of chaos, where there are multiple process flows across multiple applications involving multiple stakeholders, with no central control system coordinating it together, no single source of truth or reporting.

To address this they develop a simplified process flow and value stream map, and transposed this on to their service delivery toolchain, which featured:

- ServiceNow for ITSM.
- Azure DevOps, for CI/CD and TFS for user stories.
- Sharepoint for storing artifacts.
- iRise for ideation and mockups.
- TestComplete for automated testing.
- SQL Server for reporting repositories, feeding Tableau and Microstrategy.

Tasktop Integration Hub

They then utilized the Tasktop Integration Hub to build bidirectional synchronization for every one of the state transitions and artifact types, eliminating the challenge of it being difficult to plan and execute across teams, as all those transition points are now automated.

For example an incident being logged through the help-desk, that would create a defect in the Azure DevOps backlog. When it is updated by a developer the help-desk team can see it in real-time.

At 6m:45s Jeff walks through the details of the benefits achieved from this approach. Remarkably it delivered value within 24 hours of going live – Over 1,200 incidents were synchronized out of the help-desk, de-duplicating 671 items from the backlog, over 4,700 hours of work. A full FTE per day was saved just by eliminating the swivel chair data entry.

Implementing Flow Metrics

Select Medical also then adopted the Tasktop Viz product, which as the name suggests caters for visibility reporting, providing the tools to implement Flow Metrics, explained from 11m:00s onwards.

Jeff begins by highlighting their primary challenge was having a lot of work items in flight at any one time, and they experienced a lot of bottlenecks.

They initially suspected this was due to a shortage of developers, but Viz provided them with the insights to realize that actually it was caused by insufficient BA resources. This data was then taken to the CIO who authorized hiring of the required resource, an unprecedented event for the company, which addressed the bottleneck and sped up the work.

Viz equips teams with the tools to identify work load bottlenecks, where you can click 'Analyze Load' to identify throughputs and restraints, and importantly, by connecting to all the tools used it provides a holistic view of the entire value chain.

For example Jeff showcases a key insight: Due to their adoption of these tools they were able to increase their release frequency to once a week, but this actually caused a spike in support incidents, as too many new features too quickly confused the end users. So instead they slowed this to once a month and better managed the roll outs, with more user training etc.

This highlights that simply improving the velocity of software development throughput is not a standalone goal, but rather the delivery of new products that are successfully adopted by users is instead the objective.

Seamless adoption

At 21m:40s Jeff describes how they connected Viz with their toolchain, emphasizing a key goal was to begin Flow Metric reporting with minimal impact upon their existing work.

Part of the challenge was that Azure DevOps is a very complex environment, but they were able to easily map it's queues, backlogs and touch points as is into Viz and begin collecting metrics. It didn't require them to do anything different in Azure or ServiceNow.

From 23m:30s Nicole explains that Tasktop can model all of the flows across the toolchain, identifying key activities and attributes like debt, features, defects and flow states, such as active, done or waiting.

This last point is key because often organizations aren't even tracking wait-state work. Jeff validated this by highlighting that Azure DevOps doesn't have an On Hold status for its' user stories, which they have since remedied.

Keynote Conclusion: Accelerating Concept to Cash

A critical insight that Nicole shares is the point about the scope of the value stream mapping. Often development teams focus only on a subset of steps, notably only those that they deal with in terms of code commits, often due to what is known as a local optimization bias.

The headline goal of DevOps Flow is to view activity from a holistic business perspective, where the full process actually extends right from the very first ideation right through the end result of deployed operations.

Tasktop synchronizes workflows across all of these applications so that metrics and optimization activity is dealing with this complete end-to-end value chain.

This means performance is being managed at a level that is meaningful to business users; in short the process often described as 'Concept to Cash' – How long does it take the business to translate a new idea into a revenue-generating service.

Thus they can identify and invest in addressing flow bottlenecks within a context of there being a clearly understood and expected improvement that can be quantified in hard ROI terms.