SAFe® vs. DevOps:

Understanding enterprise frameworks for agility and value at scale

www.techtowntraining.com

{ TECHTOWN }

SAFe® vs. DevOps:

Understanding enterprise frameworks for agility and value at scale

A white paper from the Techtown lab

Introduction

We confess right away that the title of this paper is a little misleading: saying "SAFe vs. DevOps" implies that there is something mutually exclusive about the two. We don't believe this is true. At first glance, comparing a productized program management framework like SAFe and an open community-driven movement like DevOps may seem like a discussion of apples and oranges.

As we'll see, the two bodies of practice do have important differences, but in some ways they share more in common. More importantly, both carry significant implications for virtually every aspect of how we manage workflow in the enterprise, and in how we organize and empower our teams to build products and systems. We need to understand what value each approach has to offer, and how they can be used either separately or in unison as a way to transform our organizations into something more efficient, more effective, and more successful.

Contents

Is there a relationship between SAFe® and DevOps?

Do you work for a big company? Does your company have a lot of technology teams and technology-dependent projects?

Do you have a heartfelt wish to see your large organization be an employer you can be proud of – a place that fosters innovation, produces cool products, builds great teams, and isn't hampered by internal politics and confusion? Oh, and making money is also very important.

So, does that describe your organization?

It's OK if all those boxes aren't checked. In reality, before you can get to these seemingly high-minded goals, there's a more immediate and difficult task at hand for your company: staying in business.

Here's our view: In a twist of fate, what really matters is whether the people who lead your teams and make leadership decisions are committed to relentlessly working toward the higher-minded goals, because the companies who prioritize those goals are the ones who will stay in business to sell another day.

Big, complex organizations usually face big, serious challenges. The simple fact of being big means you are vulnerable. Here are a few risks that make leaders in big organizations sweat:

- You have momentum issues that smaller organizations don't face.
- You are a big target.
- You face the very difficult task of managing thousands of human beings, and coordinating their work efforts around a common vision.
- You inevitably drift towards bureaucracy – leading to blindness, waste, and a lack of institutional creativity.
- You are massively vulnerable to never-ending disruptions brought on by technology evolution.
- And so on…

And of course, because our organizations are big, the stakes are high – we're talking about the survival of our enterprise and many peoples' jobs. Unfortunately, the more noble goals of innovation, fulfilling work, healthy teams and creativity are too often considered optimistic upside that we'll indulge in thinking about some other time. But we contend – as do both the DevOps movement and the SAFe framework – that investing in these aspects of the enterprise are key to survival.

All these ideas are central to both the open, community-driven DevOps movement and the publicly-facing SAFe framework. So there is most certainly a relationship between SAFe and DevOps, and one which we think has received too little attention. In the following pages we will attempt to shed some light on this relationship.

What are the respective problems SAFe® and DevOps are trying to solve?

1. SAFe®

Why SAFe? When the Agile movement first started, it was fundamentally about clarifying how a group of human beings can work together as a team to create valuable software products in a way that works in the practical day-to-day world. The founders of the Agile Manifesto understood that the creation and delivery of software is about more than just good engineering. It's about how people work together in teams, and it's about giving other stakeholders in our projects – who may not understand the actual engineering but who do carry weighty decision authority – a frequent, meaningful place in the engineering projects upon which their fates depend. The ubiquitous backdrop of a large corporate employer certainly fed many of the frustrations leading to Agile, but the original Agile movement did not directly address this backdrop.

At the team level, these Agile principles have worked. Indeed, they have worked well enough that Fortune businesses and government agencies are ready and willing to be instructed on how they can leverage agile practices for their own success. The issue that has arisen is the fact that although an Agile practice can work great at the team level, the original Agile framework runs into big unanswered questions when it comes to adopting Agile principles enterprise-wide. To work at scale, you need more than highly-functioning independent teams. You need a framework for coordinating those teams and making upstream decisions that bring management intention to the outcomes of team projects. Once again, big challenges arise just from being big. The SAFe framework is one approach - developed by Dean Leffingwell and his group of agilists - which attempts to take the same principles informing Agile at the team level and use them to build further tools and practices which provide more thorough guidelines for large-scale, enterprise agility.

2. DevOps

DevOps is also highly colored by Agile heritage, but in a different way. Whereas a lot of the concepts in SAFe are concerned with decisions made "upstream" of software development and IT departments, DevOps arose primarily as a group of ideas intended to improve the outcomes that happen downstream of development teams, Agile or otherwise. Agile practices have worked great for individual software teams, but although Agile teams stress the importance of delivering value, a software team has not actually delivered value to the enterprise simply because their software product is "done" in most cases. It has to be tested in a way that is aligned with design intent – it has to be deployed and released – it has to be maintained – it has to be secure – it has to be evaluated in production to see if it really delivered user or customer value. Once a software product meets these criteria, the enterprise itself has a chance to realize value from the product. These concerns are the purview of DevOps, and in many cases the teams responsible for them are not software developers at all. They are system administrators, infrastructure engineers, IT support staff, testers, security managers, etc.

Large organizations almost always reach a point with DevOps when they run into difficult, tangled challenges presented by bureaucracy, silos, and departmental incentives which are optimized for local outcomes. The problem, again, is in how to scale the practice. In many cases success dealing with bureaucracy and local optimization within the IT organization simply reveals a new layer of challenges caused by the same issues elsewhere in the organization. That's the nature of a big organization. The challenge with DevOps is how to apply DevOps principles outside of Dev and Ops so a true DevOps intent is realized. That's a big part of what DevOps is all about. Unfortunately, many of the job roles associated with DevOps are often not positioned to really deal with this new layer of challenges. For example, if developers and engineers in the IT department have perfected continuous integration, seamless test automation and push-button deployments they would be considered to be fairly mature by DevOps measures. But if upstream areas of the organization are not quickly consuming feedback as a result of these practices and continuously adapting the way they prioritize and request execution from the downstream teams, then the potential of the overall value stream will never be realized.

The good news is that a framework like SAFe - or another framework for "Agile at scale," even one you invent yourself, can expand the success of DevOps, enabling more overall speed and flexibility throughout the organization. For the rest of this paper we will explain how this works. We use SAFe as our example framework because SAFe's most recent version, SAFe 4.0, is arguably the most well-developed and mature framework for realizing the goal of large-scale agility.

"What do SAFe® and DevOps have in common?"

In fact, there are a number of common denominators between SAFe and DevOps. Three of the most fundamental are:

The Agile heart

We've already touched on how Agile practices inform both the decision-making framework espoused by SAFe, and the practical delivery and operational aspects of DevOps. But let's take a moment to review exactly what we mean when we refer to the Agile heritage underlying both SAFe and DevOps. The original Agile movement clearly outlines these priorities:

- Responsiveness to emergent requirements
- Incremental development
- Continuous testing and user feedback
- Regular deployment

For people familiar with only SAFe or DevOps, but not both, it can be surprising to see that they each hold closely to these fundamental Agile principles. But it shouldn't be; one thing that has not changed in the 15+ years since the Agile manifesto was published is the original intent. What has changed is the evolution and propagation of the tools and frameworks required to actually realize these intentions at scale, in the real world.

The rise of DevOps and the success of SAFe as a structured framework are both part of that Agile evolution.

A cross-functional focus

Both SAFe and DevOps constantly stress the need for organizing teams and projects in a way that allows visibility, communication and collaboration between diverse functional players. Concurrently, never before has the need for specific functional roles been greater - it's just that functional roles have to work together instead of being batched into siloed functional departments as they typically are.

On the DevOps side, practitioners who develop software and systems cannot be assured of fast, high-quality delivery if they have not included the teams who test, deploy and maintain those systems downstream in production.

They need a shared design intent and fast test feedback so they can quickly find and fix problems while they are still easy and cheap to correct. Likewise, upstream teams who articulate business needs and requirements must have established feedback loops in place and strong cross-functional collaboration driving how they prioritize, fund and make project selection decisions. If they do not, there will be blind spots in the enterprise's overall situational awareness and it will be impossible for upstream decision makers to make sure downstream delivery realizes the original intent.

With SAFe, the recognition that Agile teams are the fundamental components of value delivery gives a nod to cross-functional importance, since cross-functional teamwork is closely associated with Agile engineering. But it's the decision to orient large and complex enterprise portfolios around epic value streams that really drives home the importance of organizing the enterprise and its teams in a cross-functional way. Value streams do not respect the boundaries of any department, function or business unit. To organize and optimize the enterprise around a value stream, we need constructs for tracing and mapping it, as well as how to prevent functional and departmental boundaries from constraining it. We also need new processes for executing work which nurtures the overall value stream and enables the delivery of value without it hinging on one particular function or department. SAFe offers guidance in these areas.

Lean principles for managing complex adaptive systems

As much as Agile has been a forerunner of SAFe and DevOps, it's the heritage of Lean that can inform us the most about how SAFe and DevOps address the problems of large organizations. As important as Agile is, Lean is possibly more fundamental to both SAFe and DevOps.

In the book that made DevOps famous, The Phoenix Project, many parallels were drawn between enterprise IT workflow and mass manufacturing work. The quality movements in manufacturing evolved into Lean, and Gene Kim and his co-authors used Lean as their analog when writing about DevOps. We consider Lean to be so important that we'll actually devote quite a bit of time referencing it throughout the rest of this paper to give perspective on both SAFe and DevOps.

A familiar enterprise

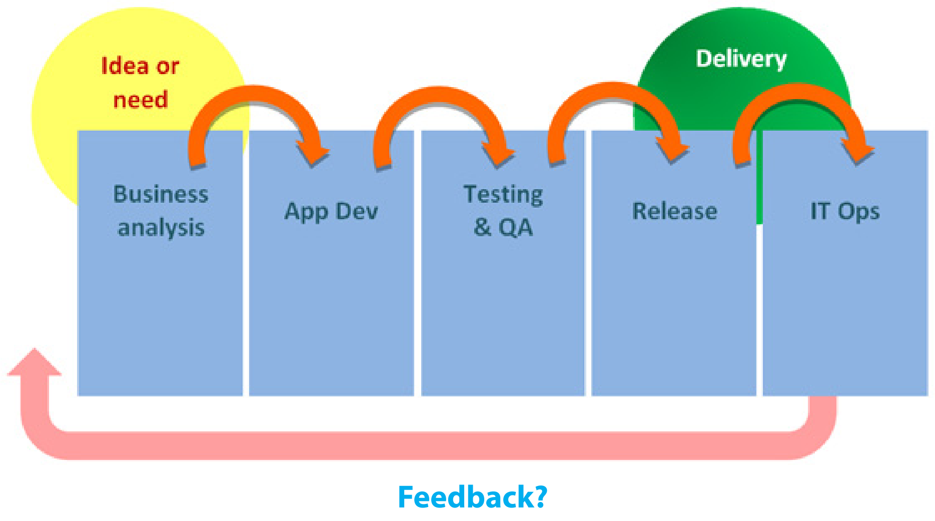

Figure 1 – Sequential, phase-gated functional batching

In most organizations, work is bundled into projects and sent through functional departments which take their turn before delivery. Between each functional department there is some sort of handoff, often with a project manager serving as coordinator. Queues form within functional departments and every project upstream in the workflow is subject to waiting on projects ahead of them in the functional batch. This type of enterprise workflow is what SAFe calls "assumptive, phase-gated, waterfall methods of the past." Indeed, there are a number of huge issues with this type of status quo.

First, PMOs and program managers are supposed to play air traffic controller as multiple projects run through the functional course, seeing to it that higher value or urgent projects are prioritized. However – because the primary organizational constructs for the workflow are 1) functional areas and 2) projects – the way functional areas are utilized play an outsized role in how project work is organized. Also, because project work is so often organized using estimates and critical path baselines, once predictive plans are laid flexibility for the value path becomes extremely difficult. It's hard to tamper with one project timeline and path without adversely interfering with other projects. Combined with the fact that critical paths are based on functional hand-offs, we begin to see why conventional PMO methodology is inefficient. How can a PMO be flexible with project plans when every project workflow is enmeshed with every other project and batched into functional areas? Even if program managers tried to exercise any flexible orchestration, it would make the lives of the people in the functional departments miserable. Of course, at the time of this writing we can see that functional workers have challenges aplenty as it is, and the challenges get more miserable the further down the stream you are in the functional flow.

Functional batching is not just a problem because it makes for brittle program management. The real problem with functional batching is far worse: it's the enormous waste caused by the waiting of every project upstream. We can easily visualize this by looking at a simple value stream map for a typical feature that's pretty small. It's immediately obvious how much waiting waste is inherent in the value stream of this project. This happens - at scale - all the time in our organizations. This likely feels familiar to anyone who has been involved in project or program management.

Unfortunately, this bird's-eye visibility is not available to the teams working in functional areas. So the best we can do is queue work and prioritize in the order received. This is problematic because higher value, and sometimes faster, projects queue up behind lesser-value projects that were in line first.

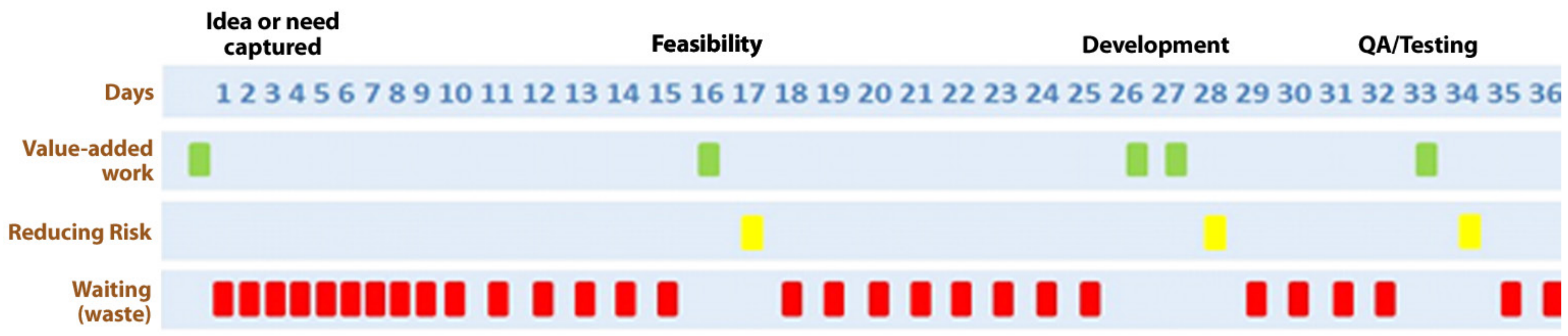

Figure 2 - Time spent on value-added work performed in an example feature's value stream

Of course, even if we could prioritize higher-value projects, we often don't have any way of knowing the relative value of projects because they are not yet delivered. We may estimate their value during planning and feasibility study, but if we don't have a light-weight economic framework for qualifying value quickly, chances are that the time passed since estimation and delivery is so great that the real-world value of the project is in question. This is exacerbated by the excessive waiting time that makes up a typical project's timeframe.

Finally, all this results in work-in-progress (WIP) piling up within each functional area. As WIP capacity nears 100% in a functional area – and when does it not? – waiting time increases for every project housed in that area and the whole mess becomes a vicious cycle of inefficient batching and waiting.

Where DevOps is having impact

In organizations where DevOps is being practiced successfully, it is having significant impact dissolving barriers and reducing hard handoffs between functional teams responsible for delivering deployed software and systems. Long established siloes between technical teams and IT groups are becoming known as sources of dysfunctional incentives which must be addressed. After years and decades of misaligned interests and little attention to how value flows are impacted by them, DevOps practices insist on closer connections and exchanges of information across previously siloed roles.

In a DevOps shop, actionable feedback mechanisms fuel continuous improvement and innovation among the teams who are closest to an organization's technology capabilities. Tooling and automation are acting as force multipliers for software engineering and delivery, product development, testing and speed. Cloud capability and an explosion of open-source toolchain patterns and social code sharing are allowing DevOps practitioners to revolutionize production and delivery of IT products and services. DevOps as a professional movement notably resembles the quality revolution in manufacturing born from Deming, realized in Japan, and gathered into the principles of Lean quality management which are now standard in the world of physical production.

Where SAFe is having impact

SAFe is just now catching on, but it is seeing rapid adoption in global enterprise organizations, sometimes piecemeal and sometimes as part of a unified transformation. Sometimes SAFe in its entirety is not the best framework or doesn't contain everything an organization needs. But SAFe is certainly driving awareness and mainstream practices which seek to build a more empowering corporate structure for project and engineering teams.

At the time of this writing SAFe is also the only published scaled Agile guide to explicitly write DevOps into its framework, although it's a bit thin on execution details. In our opinion, the greatest contributions SAFe is offering the enterprise project community are:

1. A consistent and quantifiable economic framework for prioritizing and funding features and projects on a unified backlog. We can debate whether SAFe offers the best economic framework for doing so, but the fact that it insists on one makes it a lot better than conventional ways many enterprises perform project and program decision-making. We applaud this step and encourage enterprises to place more focus and importance on front-end economic frameworks for decision-making.

2. In SAFe v4.0, the decision to orient top-level planning around value streams in the enterprise is a significant innovation. Managing large enterprise portfolios this way has rarely been done and is largely uncharted territory, but it is very exciting to think of the potential improvements in enterprise agility that might be gained by funding and organizing around the value stream instead of around functional handoffs or project requirements which are crudely predictive at best.

3. The literacy and emphasis on Lean concepts throughout all phases of the project and product delivery cycle is also a superior step forward in doctrines for project and program management. For too long lean practices haven't gotten the attention they deserve in large, technology-driven organizations.

The Key Differences between SAFe® and DevOps

Of the various frameworks available for scaling Agile from a team practice to an enterprise behavior, SAFe is the one which most explicitly addresses DevOps, but even so, it doesn't provide many details about integrating the two ideas. Essentially, SAFe recommends that you embed representation from IT operations into the Agile Release Train (ART) process. This is essentially no more than saying "make sure operations people are involved in your software development pipeline from the beginning." This is excellent advice, but no more substantial than the basic definition of DevOps in the first place.

The most important difference between safe and DevOps is that SAFe is somebody's product. Although the framework is openly published, a small industry is arising around the governance and certification system SAFe has designed around it. DevOps on the other hand is consistently defined by a loosely associated community of thought leaders and everyday practitioners who expressly stress that DevOps is a movement defined first and foremost by whatever is working for practicing professionals.

It is simplistic but accurate to say that SAFe and DevOps are different responses to distinct but related needs which have arisen in the enterprise. SAFe is intended to create a framework for cross functional Agile teams to fuel overall agility across the enterprise. DevOps gives guidance on how to apply Agile principles to the actual delivery of the IT services and products that engineering teams are responsible for. SAFe attempts to coordinate and tie the many people and teams to your overall vision, forging them into a unified engine of value creation.

Lean (the deeper common denominator)

We stressed the importance of Lean and promised we'd spend more time on it. Consider the following Lean themes, and note that both SAFe and DevOps are each a big family of ideas which purport to rest on these Lean foundations:

- Systems thinking
- Emphasizing the criticality of people and learning
- Orienting around flow of value
- Reducing enterprise waste, especially from waiting, queueing, batching and work in progress (WIP).

The SAFe framework espouses nine "Lean-Agile" principles as its guiding core. These principles, as described in SAFe, will be immediately familiar to anyone involved in DevOps. It is important to understand that Lean principles existed long before either SAFe or DevOps. SAFe has explicitly co-opted them as the underpinning of their framework. DevOps cites and implies Lean at every turn, and the more accomplished thought leaders in the DevOps movement will readily cite Lean principles as foundational to much of DevOps.

"Systems thinking" is one of the prime Lean ideas underpinning both SAFe and DevOps. SAFe lists thinking high on its list of nine Lean-Agile core principles, and in The Phoenix Project Gene Kim cites systems thinking as the "first way" of DevOps. We will discuss the other eight Lean-Agile principles of SAFe and how they relate to DevOps in a few moments, but first the idea of systems thinking deserves a deeper look.

Lean and "Systems Thinking"

Both the DevOps movement and SAFe immediately put the idea of systems thinking front and center as cornerstone principles. We want to explore systems thinking a little more here because:

1. Both frameworks highlight it so prominently
2. It's a phrase that can seem a bit vague and subject to interpretation

So what is "systems thinking," and what is meant by it in the context of SAFe and DevOps? A more practical question might be, "How do you align for value delivery across various parts of a large, complex system?"

As we've discussed, one of the biggest obstacles to scaling Agile principles lies in the fact that Agile was born as a way for teams to work. This origin lends itself to teams optimizing for local outcomes. But we've rapidly discovered that local optimization in technology organizations is directly antithetical to overall technology success at scale. DevOps is one strong attempt to de-silo development and operations teams specifically, and retool their incentives so they align with the overall mission. But DevOps founders have always realized that DevOps success relies on more than just addressing local optimization between software development and IT operations teams – it relies on financial decisions, upstream business strategy, security needs, testing/QA and many other departments.

SAFe represents an attempt to offer more formal guidance and leadership on how to address these other areas, especially the upstream decision-making concerns. So in a way, SAFe is a larger realization of the DevOps vision – de-siloing incentives and departments in a way that institutionalizes continuity of value flow, fast adaptation, continuous delivery of value, and orienting finance around value streams.

So systems thinking is a well-accepted fundamental aspect of both the SAFe and DevOps DNA. The more complex challenges lie in how to actually operationalize and execute enterprise work in a way that is faithful to this understanding. How do we manage and sync the different stages of value creation, flow and delivery throughout the many areas of the organization? Coordinating and using SAFe and DevOps in tandem is one way, and it can be a great one. But the larger point is that leveraging suitable frameworks to address different areas of the enterprise is a great approach for today's large organizations and the complexities they face as a result of large size and rapid change.

Component-based engineering (small, easily interchanged parts)

To give just one of many possible examples, consider component-based engineering. A good engineer designing a system starts with the overall purpose the system is intended to solve, then begins to reduce that purpose into smaller parts which are served by specialized components. Each component performs some smaller function which contributes to the overall outcome. For instance, in a car the engine produces raw power, the alternator produces electricity, and the radiator dissipates heat. There are subsystems for steering, cooling, computing etc. The individual quality of these components isn't nearly as important as the overall outcome of the whole system.
Besides obvious mechanical necessities of designing in this reductive way, an important result is that the system is persistently robust beyond the limitations of its individual components. If one part breaks, it may break the system but only that one part need be replaced or repaired to get the whole thing working again. In truly robust and well-designed systems, a component can break without breaking the larger system. Thus the engineer of a system can design in the ability to mitigate future adversity, or protect against disruption to a mission critical outcome. This is just the type of intention we should apply in the enterprise.

To an automotive engineer, this is all elementary. But in the enterprise, thinking about outcomes and projects this way is all too rare. Despite the fact that the enterprise is also a system in the very same way that a car is a system, we often see large organizations that don't show enough commitment to persistently thinking about how to tune their overall system to the long-term mission. Of course, many of the most important "components" in the enterprise are people and teams. We can't engineer a team of people the same way we do a piece of metal in a car, but we certainly can – and should – take the same system-level approach to engineering enterprise process and workflow.

For systems composed largely of software or IT services, this type of design approach has some profound implications, because software and IT service outputs can be deconstructed into really tiny little components. Consequently, the same is true of software and system testing processes. That's very exciting when thinking about enterprise potential. But it takes some doing to actually institutionalize working this way. It requires empowerment and enablement of individuals and teams in the organization. Large organizations can have a tendency to stifle the underlying fundamentals that create the environment for this empowerment and enablement.

Just think about how much departmental workflow these ideas impact in a typical large organization. The question is – can you operationalize them? When done well, you might have an organizational structure that enables something like microservices, which can be a really effective way to increase your technology capabilities if it's appropriate for your teams. Of course, microservices aren't going to make sense for every organization. Like anything associated with DevOps or SAFe, not every tool will be applicable, and there are no one-size-fits-all solutions. Solving unique problems and tailoring the framework to work for your unique needs are fundamental principles of both DevOps and SAFe.

Testing and understanding interactions between parts of the system

As DevOps has matured in the wild, many of the results most enterprises have hoped for have boiled down to a few common themes: empowering developers to quickly collaborate, build and deploy; continuously delivering a valuable flow of fast, frequent, incremental product value; automating the provisioning and availability of common development, testing and deployment environments. The most important enabler of all these results is component-based, automated testing. The inclusion of QA owners and testbed staff in the design process, and designing/developing to accommodate this testing framework, is one of the most important prerequisites to a truly effective DevOps and scaled Agile practice. It is commendable that the SAFe framework recognizes the importance of testing as an enabler of these outcomes. Where SAFe does not provide robust guidance is in how to actually implement, integrate and execute on this type of testing practice. DevOps, with its extensive understanding of continuous integration and continuous deployment (CI/CD) and how to use test automation as a fast, flexible, highly automated deployment framework, DevOps steps in with the knowledge and practices enterprise teams need to actually implement and execute on this SAFe mandate.

Optimizing for overall system performance, not local outcomes

The "systems thinking" ethic is a cornerstone of both SAFe and DevOps. The central problem we seek to address with systems thinking in the enterprise is how to align the components of the organization such that everything is optimized around an overall value stream, not local departments and incentives. In taking systems thinking from an abstract idea to an execution practice, figuring out how to achieve this type of orientation for your enterprise workflow is the most critical success factor.

When the DevOps movement first started, the problem of local optimization foiling overall mission was fundamental. An IT operations department optimized by an incentive to maintain stability and uptime will always resist changes handed down by an upstream development department. In turn, development is incentivized to create and introduce change into the other team's area of responsibility. With the success of Agile development, these changes get more frequent and more experimental, so the stability entrusted to IT operations comes under even more extreme threat. Nothing is more central to DevOps than dissolving this siloed condition between departments.

Often, once a development and operations department has begun to integrate and organize their work practices around holistic value delivery, we quickly discover that there are many more silos in an enterprise than just development and operations.

Enter SAFe, which insists that there is a need for alignment across all areas of the business, not just the sub-departments of IT. DevOps leaders will stress that alignment of all the teams in the enterprise has always been a central premise, but in a real-world DevOps practice we often see that most of the focus comes from within the IT department. This is what some leaders in the DevOps community have termed "DevOps Lite." Ironically, DevOps in the wild can sometimes slip into a tendency to be optimized for itself when the original impetus for the movement was to figure out the problem of upstream business alignment with downstream IT departments.

However, once you understand this you have a good foundation: if you can adopt DevOps values successfully, the SAFe framework can be a force multiplier to make DevOps more scalable and successful in enabling overall business agility. SAFe gives change agents tools and guidance on how to do so.

Continuous improvement and forward problem-solving.

It is tempting to focus on finding a specific process or toolchain to address the complex challenge of organizational agility at scale. But once again the common denominator is Lean. Beyond all the portfolio management recommendations of SAFe, and all the innovative tools associated with DevOps, lies more fundamental and human-centric prerequisites. SAFe writes the "house of Lean" illustration directly into its framework. The house of Lean is not a process framework. It is a set of values. The idea is that process engineering, product development, enterprise workflow, business agility and so forth are areas of such complexity that no specific set of tools or processes will ever be assured to work exactly the same way twice.

Indeed, it is the idea of navigating this Gordian complexity that has led to the adaptive, gradual problem solving, and sometimes experimental principles of Agile being so relevant in the first place.

Navigating complex adaptive systems

Both DevOps thought leaders and SAFe founders have adopted the stance that large organizations are "complex adaptive systems." With many different initiatives and departments, overlaid by the accelerant of rapidly evolving technology, they are not as static or predictable as traditional "phase gated" management practices assume. They adapt to all sorts of stimuli, usually without as much control as we might wish. Outcomes in the complex enterprise are often not entirely - or at all - the result of intentional initiatives. Success and failure are both subject to uncertainty, produced as much by accident as by intention. Furthermore, often outcomes, both positive and negative, are the result of unforeseen interactions between different intentional initiatives and unforeseen circumstances.

Accepting this reality is unsettling. How should one expect to navigate such unpredictability and lack of control and still be successful? There is only one solution: embrace guiding principles which prioritize exploration, experimentation, rapid adaption and continuous improvement. The technique is not complicated - it's the same scientific method we were taught in grade school. The basic idea that one can find and learn better, more innovative ways of doing things boils down to disciplined, focused hypotheses which get objectively tested. We navigate the complex adaptive environment by operating in a way which allows us to discover how to navigate it. The faster our organization is at experimenting and learning, the more it will thrive.

SAFe® Co-opts Lean, Specifying "Lean-Agile" Principles as its Guiding Values

Although the folks at SAFe did not invent Lean principles, they do a good job boiling down their framework to nine core Lean principles that precede everything else. By understanding the core Lean-Agile principles the SAFe framework describes, one can get oriented around how SAFe and DevOps can complement each other and work together. We already discussed systems thinking. Let's take a look at each of the other Lean-Agile principles of SAFe and how we can use them to compare and relate SAFe and DevOps.

1. Apply an economic view

SAFe stresses this, and because SAFe is arguably a little more focused on project, program, and portfolio management more than back-office technical execution, there's a lot more formal guidance on how to do it than there is in the loose body of knowledge that is DevOps. SAFe relies on WSJF and other decision aids to prioritize and organize technology projects. While the SAFe recipe may not always be the best one for you, it has merit in that it insists on an economic framework for technology project decisions.

On the other hand, economic decision frameworks aren't always an obvious conversation point at DevOps events, but they always should be, and DevOps thought leaders have spoken extensively about this need. Indeed, serious and successful DevOps organizations have definitely found ways to sync their engineering teams and projects with finance and have some sort of objective decision-making framework that helps reduce planning and estimation overhead, reduces "ivory tower" design processes, is responsive and fast, and takes at least some of the guesswork out of how they score the value of a potential project. They may not have called it SAFe, and it may not have worked like SAFe, but the driving needs and goals are the very same as SAFe's. An economic decision framework is also a critical prerequisite to intelligently managing and/or reducing organizational WIP.

2. Accept variability

This one is pretty straightforward in principle, because the foundation of Agile is embracing and adapting to change. But in practice, this is one of the consistently most difficult ideas to embrace – that you have to get comfortable with the fact that the unknown is constantly going to play a role in your work. You might have to change yesterday's plan, and today's plan might change tomorrow. It's not if, but when. You have to solve new little problems every day just to stay on track to execute the overall vision.

It gets more difficult when you have to operationalize this principle. Business processes don't like variability. Management doesn't like unpredictability. Traditional business processes and project management processes like to rely on stuff like estimating, projection, forecasting and planning. These all have their place and have value, but they don't always leave room for the chaos the real world throws at you. The tendency to gravitate towards a set of processes which gives upstream planners and managers a false sense of security when it comes to knowing the future is just too pernicious to be ignored.

To engineer something sustainable, you have to be able to quantify, measure, predict and forecast. But to sustainably create value, you have to bake flexibility and adaptiveness into everything. Both SAFe and DevOps are fundamental attempts to come up with practical, real-world ways to accommodate variability, unpredictability and unexpected change. SAFe attempts it upstream, closer to the planning side. DevOps is at work downstream, closer to delivery. But each framework implies – and requires – something like the other, with feedback loops passing between to keep them complementary around the larger vision.

3. Incremental creation and fast learning cycles

At every turn, being Agile is about keeping cycles of work as fast as possible and maximizing the amount of work NOT done. In a large organization, it can be downright amazing to see how quickly top-heavy processes become dominant and work in progress piles up across departments. Soon it reaches a point in which it isn't coordinated in any meaningful way. Tolerate this condition at your peril.

The purpose of fast cycle time is speed, but it's also to keep feedback frequent. Having a set cadence around when it's time to come together for feedback and adaptive planning vs. when it's time to push forward with execution is crucial, and must be decided upon in your own organization. Frequent and incremental feedback is the best way to keep stakeholders engaged, assure delivery of the correct solution and manage everyone's expectations.

At every layer of management hierarchy in the SAFe framework, and in every widely accepted DevOps practice, keeping these increments of time short and fast is fundamental. Also, instituting processes to keep the cadence steady and formalize the ceremonies that mark off the time increments is particularly important in large organizations.

Nothing is more fundamental to Lean than learning. We don't just mean learning in an individual sense, as one would read a book or take a class to learn something new, although that is important. We are talking about learning in the institutional sense, in which the feedback loops that connect critical upstream and downstream stakeholders are supported by processes and tools. In many cases there are ways to automate this feedback, and when done properly automation can be a powerful force for speed and transparency throughout the enterprise. Institutionalizing learning patterns in a way that makes feedback and feed-forward loops part of the business process yields big benefits in synchronizing and reducing friction between the diverse parties involved in the overall effort.

Particularly in the DevOps world, there are many exciting tools and use cases for how to find and amplify these valuable feedback loops in a practical way. When it comes to SAFe however, there are some obvious holes in the framework related to specific recommendations on how the enterprise can institutionalize this type of feedback-centric learning. If you are implementing or considering a framework like SAFe, we suggest taking a closer look at all the thought and tooling that the DevOps community has put into this question, and figure how to leverage these tools to support the upper management layers of your framework.

Once you have established some clarity around what constitutes appropriate increments and how to capture feedback in your organization, it's a good time to think about the next Lean-Agile principle: milestones and metrics.

4. Agreed-upon milestones and metrics

Milestones: You can't cohesively orient functional and departmental work around value streams without visible, agreed-upon milestones. Other than the general guidelines of Scrum and SAFe, there is no prescriptive formula for how to arrange milestones, nor should there be. The milestones and timeframe of your workflow will be dependent upon the specifics of your organization's work.

Regardless of how you end up collaborating to set your own milestones, it's important to keep the delivery cycle frequent. One thing we've learned from the success of Agile - and the failure of traditional waterfall project methods - is that the delivery of a prototype or milestone is always going to result in some missed expectation between the recipient stakeholder and party who produces the deliverable. This is especially true for a software product, or any system which contains a software product as a significant component. The solution to this inherent friction is to keep increments manageable and short, so that ongoing visibility and feedback is a key part of the development process.

Often, the development and delivery teams in a DevOps-style shop will attempt to reduce the cycle of delivery to such a fast increment that it is continuous: thus, "continuous delivery." In a perfect world, a continuous delivery practice represents a state in which backend dependencies like automated testing frameworks, staging environments, deployment processes and production IT infrastructure are all so standardized and robustly automated that the system can absorb changes in real time, any time they get submitted, with fast automated capabilities for rolling things back if something breaks.

Of course, this can work great for delivery of software or IT service products enabled by a good tech stack and great practitioner teams, but it doesn't always work for some parts of a complex cyber-physical system which has physical dependencies beyond just software or IT services. It is critical for the people responsible for program and project planning to understand where deliverables for the system are "soft," where they are "hard" and how the two interact.

The practice of continuous delivery also isn't as directly applicable to managing and maintaining programs and portfolios of work. You must find and coordinate between downstream workflow where continuous delivery is appropriate, and the higher layers of planning, forecasting and management at the program, portfolio and value stream levels. SAFe offers a lot of guidance in these areas, and although we may not be in danger of seeing a "continuous program management" movement just yet, the reality is that most large organizations can do much better.

Metrics: Here let's use a DevOps example, because metrics and measurement is central to DevOps. There are also specific markers associated with DevOps that are widely accepted as being descriptive of overall IT performance. From a SAFe perspective, these metrics don't necessarily inform planning or program/portfolio management. But they are critical in understanding value streams and in growing your technology delivery capability.

The DevOps movement commonly agrees that these are the metrics to track and benchmark to make sure the practice is resulting in continuous improvement of IT performance:

Lead time for changes: *The lower this number, the better*. There are several ways to track lead time for changes, but in general it should be pretty easy to agree upon a trigger to start the clock and an agreed-upon trigger for when change delivery is considered complete. Make sure these triggers are agreed upon and well understood by everyone involved, so you don't create a situation in which one party is "tracking" the performance of another without everyone getting equal voice.

Deployment Frequency: The higher this number, the better. Years of statistics collected by the DevOps Research and Assessment Institute (DORA) as part of Puppet Labs' "State of DevOps" research have shown that the highest performing IT organizations have the most frequent deploy rates. The faster and more frequently you deploy, the smaller the changes are and the less pain they inflict on the teams.

Mean time to recover (MTTR): Breakage inevitably happens, so the lower this number, the better. In high performing IT organizations, failures or outages can be so fast as to be unnoticeable, but in conventional shops an outage or change failure is often disruptive enough to be costly. Many practices associated with DevOps are explicitly intended to drive this number down: blue/green deploys, feature toggles, code-based production environments, big visible displays for system health, tiny incremental changes and the instant push-button rollbacks associated with continuous delivery all support this goal.

Percentage of change fail rates: The lower this number, the better. If a lot of changes are inherent to DevOps, then it's important to track the rate at which those changes don't add value. Obviously, the higher teams can push their success rate, the better their performance.

5. Visualizing and limiting WIP, batches and queues

Earlier we touched on the fact that one of the most crippling challenges of today's large organizations is overloading of teams and work in progress (WIP). WIP is of particular concern because it hampers organizations at every level. Individual, team, project and program layers all suffer from excessive levels of work in progress. There are many challenges presented by WIP, but the most significant from a Lean standpoint is the fact that WIP directly results in two types of waste in a system: 1) waiting, and 2) overburdening of teams.

Both SAFe and DevOps give considerable attention to addressing these problems. From the SAFe side, the constructs of program and portfolio level backlogs which are unified based on economically qualified value is one tool. The economic framework used for these estimates is another. In large organizations when the value stream layer is added to the framework it presents a third powerful tool. Finally, SAFe endeavors to coordinate cross-functional teams and value streams in way that better prioritizes value and reduces functional batching.

As we can see, the majority of SAFe's focus falls on front-end planning, prioritization and coordination of work. On the delivery end of the value stream, DevOps has a somewhat more detailed guidance to offer than SAFe. Many DevOps tools and practices support the Lean idea of "one-piece flow" over functional batching. This is achieved by orienting cross-functional teams around fast, high-value outcomes and robust risk mitigation strategies. Visualizing work at every opportunity, continuous integration and delivery, test-driven development, and automation up and down the toolchain are all practices associated with DevOps which are intended to reduce batching and handoffs between functional teams, and thus WIP.

6. Cross-domain planning and cadence

We've already discussed why cross-functional perspective is so important to Lean management in a large complex environment. Value streams take all sorts of paths through departments and functional roles. Product outcomes depend on working together across functions.

The fact that SAFe has called out a Lean-Agile principle for formalized planning across domains shows a laudable understanding of how critical it is to encourage information and work flows between different areas of the business. Mapping the cadence at which this happens to key business indicators can serve as a valuable way to gauge how well or how poorly internally imposed boundaries and constraints are hampering the creation, flow and delivery of value.

The ability to orient and optimize enterprise effort around an outcome, and not a function, is the heart of Lean. This is how Lean attacks the matter of scale - by targeting a future condition without prescribing specific methodology beyond problem solving and improvement. Of course, all the teams and roles in the enterprise play an integral part in moving towards the outcome. SAFe orienting around value streams and product flows are their strategies for doing this at enterprise scale. DevOps plays a critical tactical component, and if adopted correctly DevOps practices can "close the loop" of feedback to provide the tactical data that SAFe's many managers and decision makers need to unify the cross-functional work of the organization.

7. People first: Unlock intrinsic motivation

Lean principles prioritize driving out fear and uncertainty about working conditions and employment. These are pervasive in today's enterprise organizations, and are direct results of the dysfunctional conditions which we have discussed here. Overburdening, poor coordination of projects, watching projects get cancelled, misaligned incentives and poorly managed change are all highly demoralizing to the teams impacted. They are also often consequences of the enterprise challenges we have presented, and are main areas of concern for both DevOps and SAFe.

A key Lean management priority is the removing of barriers to pride of workmanship. This is because talent and innovation arising from the doers of our work are stifled by conditions such as these. How often have we seen our most talented engineering teams crushed by stressful working conditions requiring long hours and frequent emergency heroics which receive little positive recognition?

Management layers in the business have a responsibility to address these issues. They should figure out how to welcome and encourage curiosity, experimentation and problem solving. Asking our people to solve interesting problems and create valuable solutions, and giving them visibility and direct responsibility over the successful outcome of overall initiatives are far better ways to realize value from our work than asking them to blindly work within a functional silo. We should invest in the ingenuity of front-line employees and engineer our governance and business processes to enable it.

8. Decentralized decision making

The ability to drive as much collective decision power down from central decision sources to teams of doers and executors is a key enabler of faster, more valuable flows of work. Both DevOps and SAFe encourage the organization to build well-designed frameworks which allow teams to make decisions without needing constant approval but which are also aligned with the overall mission. With SAFe, program and portfolio managers are given economic guidelines based on the needs of the business which allow these decisions. With DevOps, common processes and assets are standardized across functional teams and combined with automation to form a common library of resources for building, testing and deploying systems. DevOps practices can also be used to standardize and automate change processes.

Practices like these are powerful tools when it comes to aligning and decentralizing decision making in the enterprise. The dividends paid are increased capacity for moving work forward, much greater speed, and greater empowerment of individual teams and contributors.

Conclusion: SAFe® and DevOps both apply Lean thinking to the flow of value in the enterprise, but in different areas and in different ways.

The real winners at an enterprise level are those who can integrate and apply all that's best from both SAFe and DevOps. And here's one final common denominator: both SAFe and DevOps are openly available frameworks which are not formally governed. Each is intended to present useful recommendations which can be adopted and customized in a way that makes sense for you.

Upstream value vs. downstream value

DevOps addresses, at a practical level, much of the tooling, process and engineering concerns around building and delivering software and systems. The closer you get to delivery, the more DevOps is relevant.

As you move "upstream" from DevOps practices, closer to the fuzzy front end of planning, design and finance, the implications and questions raised by the DevOps practice about things upstream become some of the most valuable indicators for informing how you approach a framework like SAFe, or any other "scaled" Agile practice.

SAFe addresses, or begins to address, a lot of the factors which often prevent success with DevOps because of upstream decisions which are out of control for DevOps/IT engineering teams.

Conversely, although SAFe gives a lot of attention to upstream decision making about the development of working software and systems, it's too common for SAFe to feel like it defines downstream "delivery" of value as happening when software teams deliver working increments of software. This is problematic because it does not adequately account for the importance of IT operations, security, administration and database teams which are assumed to deploy and maintain those products. It is well-known in the DevOps community that smaller, more frequent "delivery" of software from Agile development teams to IT operations teams is one of the great headaches for anyone who touches the actual delivery to customer or user in a way that generates value. DevOps is a body of principles and practices, enabled by tools, which is expressly intended to resolve the challenges raised by this aspect of the enterprise workflow.

Avoiding dogmatic thinking and keeping an open mind

It's easy to fall into dogmatic thinking about a framework like SAFe, which is heavily promoted by a specific organization and has many seemingly quantifiable job roles and certifications associated with proprietary framework. DevOps, which is not a proprietary framework, prefers instead to leave itself open to whatever innovations, tools or practices work in a given use case. But dogmatic thinking can still creep in.

Of course, the more success stories we see from DevOps and SAFe, the more we see that whatever has worked in the enterprise environment almost always aligns with long-established Lean body of knowledge. The Lean principles discussed here are inherent in both SAFe's "Lean-Agile principles" and DevOps intrinsic "systems thinking." At heart both espouse Lean as guiding values.

It's important to remember that for all the tools, processes and frameworks presented by SAFe and DevOps, the heart of Lean is not any specific tool or process. The heart of Lean is a mindset dedicated to finding and solving problems so we can continuously improve. The heart of Lean is relentless, scientific discovery and solving of problems. It is human curiosity and creativity, nurtured and directed at practical application. It is a steadfast commitment to long-term quality outcomes at the expense of short-term leverage or cost savings. These qualities are the keys to both SAFe and DevOps as well, and they are completely at odds with any type of dogma or authoritarian management style. If a process or tool does not have a connection to this mindset and these commitments, it should raise a red flag that it is not actually moving the enterprise forward. This litmus test should be ubiquitous in a Lean mindset, and both SAFe and DevOps as well.

We hope this investigation of SAFe and DevOps leaves you more informed and aware of what each has to offer, and how they relate to each other. We would love to hear your feedback and perspective on the topic as well.

Struggling to Scale Agile or Implement DevOps?

If you are having challenges scaling your organizational agility or really getting started with DevOps practices, Techtown can help. We offer a deep bench of consulting experts, trainers, and coaches in these areas. Our extensive curriculum of training gets your teams up and running quickly with both the newest technology tools, and the business practices your teams need to be successful in todays high-speed, fast-change environment. For organizations who are further along, our coaching practice gives you access to the best minds in the industry to help mentor your teams and offer guidance on your strategy.

References and Further Reading

We read these books so you don't have to!

[1] Scaled Agile, Inc. *SAFe 4.0 Introduction: Overview of the Scaled Agile Framework for Lean Software and Systems Engineering.* Boulder, CO: Scaled Agile, Inc., 2016.

[2] Jez Humble, Barry O'Reilly and Joanne Molesky. *Lean Enterprise: How High Performance Organizations Innovate at Scale.* Sebastapol, CA: O'Reilly Media, 2014.

[3] Jez Humble. *"Why Scaling Agile Doesn't Work."* Presented at GOTO Berlin 2015. Berlin, 2015.

[4] Joshua Arnold. *"Managing Queues in Product Development."* Black Swan Farming, accessed January 2017. www.blackswanfarming.com.

[5] Gene Gim, George Spafford, and Kevin Behr, *The Phoenix Project.* Portland, OR, 2013.

[6] Don Reinersten, *The Principles of Product Development Flow: Second Generation Lean Product Development.* Redondo Beach, CA: Celeritas, 2009.

[7] Brian Cronauer, Andreas Huber, Thorsten Keuler. *"Agile Doesn't Scale…Without Architecture."* presented at SATURN 2012. St. Petersburg, FL, 2012.

[8] W. Edwards Deming. *Out of the Crisis*. Cambridge, MA: MIT Press, 2000.

[9] Manifesto for Agile Software Development. *Manifesto for Agile Software Development* accessed January 2017. www.agilemanifesto.org.

[10] Jez Humble and David Farley, *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*. Boston, MA: Pearson, 2011.

[11] Eric Ries, *Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. New York: Random House, 2011.

AUTHOR: Chris Knotts
DevOps and Enterprise Innovation Curriculum Director
PMP - Project Management Professional, PMI

Send your questions and feedback to info@techtowntraining.com