September 2017

DevOps INSTITUTE

Whitepapers

The Convergence of Scrum and DevOps

Dave West CEO Scrum.org, Jayne Groll CEO DevOps Institute

"Software is eating the world."[1]

Software is pervasive; it influences every aspect of our modern world. It enables organizations to deliver better products, faster and with higher quality. It enables organizations to create new, smarter products that deliver better customer outcomes. It also enables organizations to better understand user needs to become better at delivering better outcomes.

Software is also the source of disruption. Organizations who can use it to adapt and evolve faster out-perform their non-digital counterparts.[2] The formula is simple, yet hard to embrace: organizations that harness the power of software master rapid empiricism, the ability to quickly form hypotheses, try them, gather and analyze the results of these experiments, and use the information to improve.

The world is changing at an increasingly faster rate, driven by the increased velocity of technology, markets and climate change.[3] The only way to respond, survive, and thrive is to embrace the change and become better at turning it into an advantage.

Two Movements, One Goal

Agile and DevOps are the primary means by which organizations are driving these changes. They share much in common: faster delivery cycles, smaller increments (or batches) of releases, using feedback to improve, removing waste and impediments. Their emphasis varies, and this sometimes leads people to believe that they are different things. Agile emphasizes team interactions, culture, and values, while DevOps emphasizes delivery pipelines and flow. But, Agile is also concerned with automation, and DevOps is also concerned with communication and culture.

[1] Andreessen, Marc. "Why Software Is Eating The World." The Wall Street Journal. August 20, 2011. https://www.wsj.com/articles/SB10001424053111903480904576512250915629460.
[2] "2017 State of DevOps Report." Puppet. https://puppet.com/resources/whitepaper/state-of-devops-report.
[3] FRIEDMAN, THOMAS L. THANK YOU FOR BEING LATE: an optimists guide to thriving in the age of accelerations. S.l.: PICADOR, 2017.

Both aspects are essential. Both communities have great ideas and abundant enthusiasm. Organizations need both to succeed; to the degree that they see them as separate things, or focus on one at the expense of the other, they fail.

It is almost ironic for anyone to say that Agile and DevOps are opposed to each other or even shouldn't be thought of together. The most popular form taken in delivering software with greater agility is Scrum, some estimate over 18 million software professionals [4]practice Scrum every day making Scrum and therefore Agile a key component of how teams deliver software and therefore at a minimum the "Dev" in DevOps.

Both movements pursue the desire to deliver value to customers in a more effective way, but from different starting points; Agile with the developer, and DevOps typically from operations. These apparent differences are amplified by misunderstandings including:

- Some Agilists think ALL process is bad, while Ops is steeped in a history of stability through process.
- DevOps looks to tools and automation to increase speed, while some Agilists view tools as a necessary evil at best, and a distraction at worst.
- DevOps looks to holistic systems thinking to solve problems, while Agile looks to people to drive change and discover it through work.

Politics and Organizational Silos Widen the Gap

Lasting change requires strong leaders. They provide not only clarity and direction, but provide incentives and permission to change. Organizational change also attracts strong personalities who get recognized for being a force for change, and who get rewarded when those changes produce results. When their success is tied to the success of that program, competing programs are a threat. When Agile and DevOps initiatives are led by different parts of the organization, competing political goals lead to confusion:

- **Different tool strategies being adopted between the competing initiatives.** How many times have you heard, "the agile teams use JIRA, but the ops teams are using Service Now"? Those tools and the vendors selling them focus on a change initiative to drive purchases. This makes it harder and harder to integrate those groups of people as their data and processes are hidden within different tools.
- **Politics rather than value being used to make decisions.** The saying, "it is not what you know, it is who you know," is not just true in national capitals, but in any organization, that has grown larger than 20 people. Politics is the reality of most organizations and that means that the associated initiatives not only come with baggage of the work but also the politics of who is driving them.
- **Increased division and organizational conflict.** Ironically, the DevOps and Agile movements call for unification and collaboration between different groups but often, when practiced, lead to even deeper gaps with traditional groups because of their different terminology and

[4] https://www.infoq.com/news/2014/01/IDC-software-developers

ideas. That is also evident in the event, publication community with VERY different shows, magazines and blog sites being visited by members of these communities.

Development and Operations Have Different Cultures

Agilists can seem dogmatic to DevOps professionals. "Oh, you are one of those Agile people," is a phrase often used to describe the growing resentment that developers and other IT professionals feel towards the Agile community. This resentment is perhaps driven in part by the success of Agile which has led to large numbers of Agile coaches, some who are fantastic, some good, and like with any mass number of people, some not so good.

Also, agility is often undermined by the organization itself, which reduces credibility and success.

DevOps professionals can seem too process and tools focused to Agile proponents. The slogan, "don't do Agile, BE Agile," is used to describe the people-centric focus that Agile encourages. Instead of complex, detailed processes, Agile would encourage organizations to provide a lightweight framework, such as Scrum, and then add any process the team feels adds value. Agile proponents consider process to be "emergent" in response to the needs of the situation. Teams are empowered to make decisions about how they work. Operations traditionally takes a different tact with detailed checklists, process controls and tools. Change, after all, brings instability and needs to be appropriately managed.

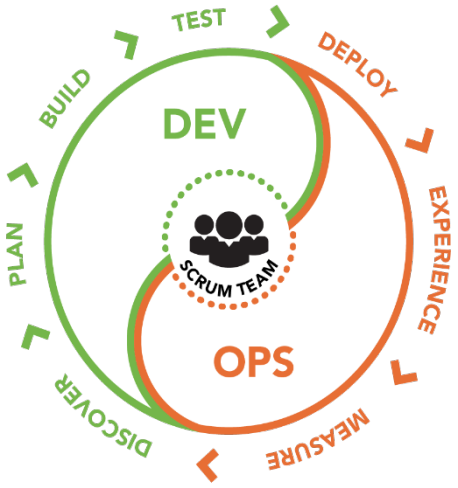
Language does matter, and teams need consistent language and concepts to work together. The better teams work together, the better the work and value delivered. When communication breaks down, collaboration suffers, which leads to:

- **Alienated stakeholders.** Business partners and managers see the conflict and lose confidence that anyone knows what they are doing.
- **Culture clash between Dev and Ops.** The "touchy feely" people-oriented Agile community is very different from the process-oriented IT operations group.
- **Increased fear and defensive behavior.** No one wants more risk but they respond to it differently. Agile approaches reducing risk by making lots of small but low risk changes, but Ops sees lots of change as inherently risky and responds by increasing process, control and governance.
- **Developers view DevOps as a stop-gap; they see the goal as "NoOps".** NoOps refers to complete automation of Ops work, allowing developers to run the production environment. Whether this is possible or desirable is beside the point; most organizations are so far from effective DevOps that NoOps is a vague and unhelpful goal.

The reality is that delivering value to customers through software includes elements that require detailed, comprehensive process, and other elements that are fluid and require the team to work in a dynamic manner. Automation is the ultimate stable process, but for many situations automation is not possible, or the cost outweighs the benefits. Balance between automation, process, and empowered teams is essential. If you focus on any one and ignore the others it leads to:

- **Poor quality.** A systematic approach to quality is crucial for the quality of any product, but for many teams, quality continues to be spotty at best with manual testing surrounding their continuous integration process.
- **Never getting anything "really done".** By having too many complex processes that require external validation or verification, it is almost impossible for a team to deliver a backlog item to Done.
- **Lack of transparency and visibility.** Many processes are so complex that the intent is lost and it is impossible to make balance-based decisions.
- **When the team changes, everything changes.** Without a clear, defined process it is difficult to manage changes in people or have any chance to scale.
- **Work is hidden because the process does not explicitly support it.** As processes become more detailed and complete, outlying work falls through the cracks. This either leads to lots of 'hidden' work and 'work-arounds' or the work is missed.

Bringing Together Scrum and DevOps



Imagine a world where cross-functional, business-oriented teams deliver working software continuously and where work flows seamlessly between all the right stakeholders in real time. Add to that where value is clearly understood, measured and reported on. This vision is the reality for many smaller startups that have blended business and technology with customer and market to work in a holistic way of delivering customer value. Scrum Teams are accountable for delivering software and operations teams are responsible for putting the environment in place that enables them to do it in a secure, safe, high quality way.

Figure 1. DevOps Cycle

Cross-Functional Teams With All the Right Skills Deliver "Done" Software, Faster

Building cross-functional teams is nothing new. Hirotaka Takeuchi and Ikujiro Nonaka described the importance of flexible, cross-functional teams in "The New New Product Delivery Game" in 1986.[5] But, making the right decisions about who should be in or out of a team is difficult. Modern products comprise a dazzling array of different technologies that historically have required specialist skills and infrastructure to support. And, a delivery team must have all the skills necessary to deliver a working product. This is described in the Scrum Guide[6] as:

"Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team."

[5] Nonaka, Hirotaka Takeuchilkujiro. "The New New Product Development Game." Harvard Business Review. August 01, 2014. https://hbr.org/1986/01/the-new-new-product-development-game.
[6] The Scrum Guide™. http://www.scrumguides.org/.

Cross-functional teams need to have the right skills to get the work done. When a team is transitioning from a traditional role-based approach, the members need to broaden their skills. They also need to build empathy for the customer to help them build better products. This doesn't mean that an individual can only be on a single team. There are many instances where someone with operations expertise can sit on multiple teams (acting almost like a "service" to the team), bringing their expertise and knowledge to the team(s), enabling better delivery while helping to educate other team members so that they can broaden their skillsets. But, that has to balanced with how their availability effects the team's ability to delivery product backlog items. They also typically need to:

- **Reduce the number of hand-offs.** Hand-offs increase wait time and reduce ownership and accountability. Hand-offs may be unavoidable when team members lack all the skills or the authority they need to deliver a complete solution, but if these account for most of the work being managed by the team, then those people should be part of the team.
- **Develop all the skills they need to deliver value.** When teams don't have all the skills they need, they will need support from specialists. Over time, they should strive to learn from these specialists so that they increase their independence. For example, all applications need security to be considered from the start. A team can be supported by the security community from whom they can learn, reducing their dependence and improving the quality of the work.
- **Increase the automation in their processes.** Automating the many mechanical tasks teams need to perform to deliver software frees them to spend more time on customer-oriented value work. Examples include continuous integration, testing, deployment, and provisioning environments, to name a few.

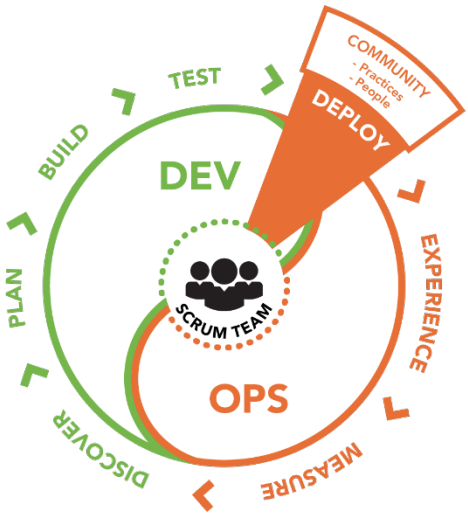


Figure 2: Deployment Composite Example

DevOps and Scrum Succeed When Impediments Are Removed

Queues break agility because they remove the ability of the team to be in control of the situation. When support organizations are connected to Agile teams, via queues, it is very likely that the team as a whole will be less effective as they wait for their item in the queue to reach the top.

When building the future delivery organization, you should incrementally remove queues and replace them with self-service capabilities. Traditional IT functions such as environment provisioning, database, security, network and even deployment functions should be developed and supported as a series of self-service capabilities. These capabilities will look much like the services from companies like Amazon Web Services (AWS), Microsoft Azure and Google, and often will take advantage of these external cloud services, adding the organization-specific intellectual properties to make them company compliant.

Legacy applications can also kill agility because, over time, these legacy systems have grown so complex that no one understands them. This complexity makes it very hard for a single team to have all the skills necessary to deliver value. For example, the amount of work and skills required to produce an end to end integration test makes it almost impossible to include in one Sprint! Unfortunately, there is no secret in solving this problem and instead requires organizations to re-architect these systems to enable more modular development, deployment and testing.

Helping grow team capabilities means:

- **Empowering teams to "do" what needs to be done to release the product.** Instead of creating large amounts of red tape to protect the team from themselves, empower teams to make decisions and deliver the product. But, ensure that the amount of change is small enough that it can be easily backed out of, coupled with environments and architectural practices that are robust enough to handle change.
- **Reducing external dependencies through self-service automation.** Rather than connecting the agile team to complex processes, provide self-service capabilities that allow the Scrum Team to drive their own outcomes. Examples include provisioning virtual machines, or getting access to test data.
- **Supporting teams with shared technical services, available without queuing.** The simpler the software is, the more Agile the team can be in developing and supporting it. As the complexity and value of the software increases it becomes harder and harder for any one team to have all the right skills to support it. But, any dependency on external skills can become a bottleneck. By concentrating on supporting Scrum Teams with experts who can coach or mentor teams on a particular technology or approach, you both remove a bottleneck and increase the skill of the team, making them more flexible and Agile.
- **Simplifying and modularizing legacy applications.** Look to refactor the legacy systems and supporting infrastructure such as environments and tests to better support Agility. That means focus on reducing the overhead of testing, provider better support for decoupling, and spread legacy knowledge throughout the teams rather than leaving it in one specialized area.

Automation, Automation and More Automation

Continuous Delivery is one of the most important sets of DevOps practices. At its most mature practice, this means that every time a developer commits code, it gets built, tested, and deployed (if it meets all release criteria) with little to no manual intervention. Not only does the lack of human beings make for faster release processes, but it also increases quality by removing the chance of human error.

The first step to automation for a team is reviewing the process for waste and non-value added work. At the same time, they also review the application architecture. The result may be new backlog items focused on refactoring and technical debt. Some good tips for automation include:

- **Simple processes are best.** Remove complexity and over-engineered processes.
- **Start from the back and the front and work toward the middle.** Unit testing and release deployment areas are perhaps the best places to start providing numerous opportunities to create scripts.

- **Automation should be part of the natural work of the Scrum Teams.** It is easy to think that there should be a separate team building automation and creating standardized infrastructure. Experience shows that these centralized tools teams tend to build for everyone and serve no one. Focus on each Scrum Team building the right stuff for their own use. These teams should work with operations to ensure that anything they build is usable in both deployment and production.

Software Delivery Professionals

At the heart of any modern delivery organization are skilled professionals that can apply an empirical process and have the engineering/operations skills to ensure that the work they do is of the right quality.

Scrum believes that professionalism is so important, that the Scrum Guide includes the five values of focus, openness, respect, commitment and courage.

These values can help an organization think about their delivery capability in a holistic manner, adding a different dimension to the delivery process. There are several tangible ways professionalism can be encouraged in a modern delivery organization:

- **Introduce an apprentice, journeyman, master model to personnel development.** Based on the medieval model of knowledge transfer, this approach to career development creates a community of learners and teachers consistent with lean thinking, where leaders get out of the conference room and the office to work with the team in Gemba, or the place value is created.
- **Reward teams, not individuals.** There are few things in most organizations that a lone individual can achieve. Many different people must work together to deliver better customer outcomes. Individual incentives are misplaced, and individual rewards ignore the reality that no individual achieves very much purely on their own.
- **Reward people for creating value, not merely keeping busy.** Focus on improving customer outcomes and removing waste, not on keeping people as busy as possible. While labor is expensive, no one benefits when busy people deliver the wrong solution. And keeping critical people with scarce skills as busy as possible often simply means that many others will end up waiting.
- **Make values as important as the outcome.** How many times have you heard someone say, "Well, he delivered, but it was a blood bath"? That heroic approach to product delivery works well occasionally, but is unpredictable and ultimately destroys the organization by burning out people and creating large amounts of technical debt. By encouraging professionals to live the Scrum Values[7] of focus, openness, respect, commitment and courage, you create an environment that not only supports agility, but is a place you would like to work. If these values are shared by development, operations and the business teams,

[7] A great overview of the Scrum Values can be found here: "There's value in the Scrum Values." Ullizee. May 03, 2013. https://guntherverheyen.com/2013/05/03/theres-value-in-the-scrum-values/.

trust is natural. Ultimately, by focusing on shared values across the organization, you build quality products, not just in terms of defects, but in terms of customer values.

Integrate, Focus, and Win

For many, the barriers to becoming an integrated, collaborative, cross-functional product delivery and support organization is almost insurmountable: the status quo seems immovable, legacy applications seem unchangeable, and the culture seems unshakeable. Some organizations' attempts to change result in little more than giving new names to old processes, roles, and organizational structures.

But a pragmatic step-by-step approach to both Agile and DevOps can provide a clear path forward while delivering benefits along the way. Not all parts of the business require the improved agility and flow afforded by Scrum+DevOps. Thus, incrementally take those parts of the business that would benefit and change them.

Changing an organization sounds like a daunting task, and it would be if one had to change an entire organization, all at once.

Change starts with a customer who could be better served, and a product delivery organization that wants to make them more satisfied.

Step 1: Integrate Your Approach to Delivering Software

This sounds simple, but many organizations find their approach is fragmented into specialist groups who are aligned around reducing cost and risk, instead of aligned around delivering customer value. Work flows from a planning function into development, test, release and then maintenance. Development teams and support teams are separated and projects will even put different groups of people working on the same systems and business areas.

Thus, the first step encourages the organization to stop this madness by taking a holistic product-centric approach to delivering business value. Projects may still exist but are secondary to persistent teams – that includes operations – that add new business value while supporting maintenance needs. Large products delivered by teams of teams would use the Nexus™ framework[8] for scaling Scrum to organize the Scrum Teams, while small products would be delivered using Scrum.

What this means for Scrum Teams is:

- **Building teams that are aligned to products and the customer.** By aligning to a product or customers, that means that the Product Backlog will include new and maintenance requests. If the product is large, these teams will be organized around a Nexus which facilitates multiple teams working on the same Product Backlog.

[8] "Scaling Scrum with Nexus™." Scrum.org. https://www.scrum.org/resources/scaling-scrum.

- **Including the right people to deliver Done software.** And when we say Done software that means into production and supported. By creating a "complete" Definition of Done[9] for the software, that means all the right people need to be involved. Operations, security, data and other groups will either be included in the teams or they will empower the Scrum Teams to undertake the tasks that would traditionally be theirs. This requires these organizations to change from doing the work in a silo to doing it as part of the team while helping others do the work and then ensuring that the work is right.
- **Taking responsibility for the complete value chain, not just the development bit.** A fundamental change for most Scrum Teams who still live in a 'Water-Scrum-Fall' world, to be effective and deliver real business value, they must be clearly responsible for the end-to-end process of delivering value as a team, breaking down the silos of ownership and individualism.

What this means for Operations is:

- **Restructuring the portfolio around products/customers.** The first step is approaching work on items in the context of the business, and restructuring around business outcomes and products, not around processes or specialist skills.
- **Removing queues from the system.** When a Scrum Team is not able to do the work themselves, don't make them wait for someone else to get help; make the help available whenever the Scrum Teams need it. Provide self-service capabilities wherever possible, or staff the "expert" positions with enough people that there are enough of them to help teams whenever they need it. Either way, a strong focus on removing impediments to flow is crucial.
- **Provide Operations Services to the Scrum Teams.** Having separate development and operations teams may be inevitable, but have team members available to be a service to the Scrum Team. Take part in the Daily Scrum, be assigned to a set of Scrum Teams, understand what is in the Product Backlog and Sprint Backlog. Be embedded with the teams you are a part of so that you can help them prioritize as much as you prioritize yourself.
- **Automate, automate, and automate.** For many operations teams the only way to trust the Scrum Team is to either be part of the team or automate the work so mistakes don't happen. To this end, automation plays a vital part in ensuring that a holistic approach to delivery is practical.

Step 2: Add Feedback Loops

Once the teams are better integrated and the work is flowing, with regular product-based retrospectives and improvements, the next step is to focus on instrumenting the processes to gain better insight into both releasing and supporting software, and the business that the software supports. Adding operational and lean/business analytics to the process provides a foundation for the Product Owner to make the right decisions about their product. It also equips the development + operations team to gain better insights to their working practices, allowing for improvements.

What this means for Scrum Teams is:

[9] "Definition of "Done"." The Scrum Guide™. http://scrumguides.org/scrum-guide.html#artifact-transparency-done.

- **Releasing software more frequently during Sprints.** The Sprint Review is NOT a phase gate; software can be released as fast as required to production, and many times throughout a Sprint. Improved feedback is not just about instrumentation and capturing more data, it is also about capturing data more frequently. If the team or Nexus is delivering software frequently, then the Sprint Review is far more interesting than 'potentially shippable', allowing the review to be on software that is already in production.
- **Building success measures into Product Backlog Items.** By looking at Product Backlog Items in terms of how success could be measured not only ensures that when the team implements the item they will have some instrumentation in place, but it also validates the item itself. Empirical process calls for inspection and adaptation based on transparency. This requires that each outcome be measurable to allow for such inspection and adaptation.
- **Making Sprint Reviews more visible and transparent to the business.** Data is only valuable when the teams are using it to build understanding and allowing decisions to be made. It is important for the Scrum Team to radiate information to the stakeholders who can help them make the right decisions. The Sprint Review is a regular, formal meeting that allows the Scrum Team and other interested parties (e.g., Stakeholders) to inspect the Increment (the work they have done that Sprint which could be a single release, many releases, or maybe just releasable but not into production). By adding metrics to the review, you can focus the review on the things that matter.
- **Instrument software to gain better understanding.** Underpinning the whole process is the need to add instrumentation to both the process of delivering software and the actual software itself.

What this means for Operations is:

- **Putting in place environmental support for instrumentation and analytics.** Environmental and application analytics require a foundation of technology to support both the capturing of information and also the storage and analysis of that data. By providing these environments and the skills necessary to understand the data, operations provide a valuable function for the Scrum Teams.
- **Building shared dashboards that show success in delivering customer outcomes.** Too often, operations and development groups are using different dashboards to report progress, success or failure. By putting in place a dashboard that is consistent for both groups you build a communication environment that allows effective collaboration.
- **Being part of the Scrum Team(s), Daily Scrum, Sprint Review and Sprint Retrospective.** This sounds very simple, but without focus on working with the Scrum Team(s), too often the operations professionals are left out of the review process. Scrum is an empirical approach; this requires constant testing of ideas through transparency, but tests are fundamentally flawed when the right stakeholders are not in the room. By making operations professionals a part of the team and involving them in the various Scrum Events, insights can be gathered, allowing the future work in the backlog to be refined and plans changed.
- **Ensuring Sprint Retrospective issues are taken into any DevOps initiatives.** There are many places that inspection and adaption happens within the Scrum framework. The Sprint Retrospective is a key event for inspecting how the team is working. Out of this

retrospective comes opportunities for improvement. Many of those improvement areas are related to the processes of release management, security and operations. It is therefore important that the operations teams are a part of ongoing Sprint Retrospectives and take them onboard as part of the improvement work.

Step 3: Grow a Culture of Continual Experimentation and Learning

The ultimate end goal is to no longer have a separate operations and Scrum Team, but instead have Scrum Teams and a community that supports them.

This might sound like semantics, but this model implies that work is only done in the Scrum Teams, and community is, as the name implies, a collection of skilled professionals that can work with, support and mentor the Scrum Teams to deliver great software. Each Scrum Team has a direct connection with the customer and, as directed by the Product Owner, continuously delivers valuable software. The work of the team is based on clearly measurable outcomes with each "experiment" or Sprint working toward learning. Supporting these Scrum Teams is infrastructure and a community that is lightly coupled to the work via self-service capabilities, team members and mentors who help the team deliver work and improve. To do this, teams need to:

- **Put in place evidence-based measures to enable teams to improve.** Evidence Based Management (EBMgt™)[10] provides an approach that looks to three groups of value-based measures in areas of current value, ability to innovate and time to market. Viewing improvement in these three dimensions helps teams focus on where they most need to improve.
- **Empower the Product Owner to implement changes and make product decisions.** No need for complex reporting and sign-off committees. The Product Owner is accountable for not just the development of the product, but also its production usage. That accountability means that operational activities are part of their mandate and need to be considered in the context of their role. Total Cost of Ownership/Operation is no longer an abstract measure that is included in a dashboard somewhere, but the reality of every decision the Product Owner is making.
- **Include "experimental" backlog items.** Not everything will be an experiment; the backlog will include work that is driven by a clear future looking hypothesis, work that has clearly defined value, bug fixes and the resolution of technical debt. Each backlog item will drive work within the Scrum Team and their support shared service/community groups. This work should be balanced, allowing teams to build for today and the future, while also allowing them to deal with the past.
- **Create a shared incentive model that aligns team members and stakeholders.** Dissolve the distinction between development, operations, and "the business" by introducing a single incentive model for everyone that rewards delivering customer value.

[10] "Evidence Based Management (EBMgt™)." Scrum.org. https://www.scrum.org/resources/evidence-based-management.

Summary

DevOps was born from the pain generated by Agile teams and was initially termed "Agile Infrastructure". The intent was to better support Agile teams with more flexible approaches to infrastructure. That intent has become part of a broader movement that is focused, not just on infrastructure, but driving a revolution throughout the IT profession. But, it is important that DevOps does not forget its roots. Supporting customer-centric Scrum/Agile teams is still fundamental for success. And that means radical changes to how change is managed, services are provided and teams are organized. There continues to be a need for the DevOps movement and Scrum Teams to work together to deliver DONE software, and support each other in their constant need to improve by inspection and adaptation through transparency.

Organizational Models

There is no one-size-fits-all model for how operations professionals connect to Scrum Teams. In some situations, operations will be embedded within the teams. In other situations, operations professionals will be a shared person or people that work with and mentor multiple teams in their work. For more complex products that are using a Nexus approach to organize teams of teams, operations may have representation within the Nexus Integration Team (NIT), as well as being part of the individual Scrum Teams. What is true is that each organization will have multiple models running because their products/customer situations will require different support needs. And, organizations that think that optimizing the efficiency of operations is more important than delivering value will never become Agile. When considering how operations is oriented, consider the following:

- **Does the amount of complexity and the unknown make any shared model impossible to operate?** In the case of situations where there are many unknowns around the operational nature of the solution, putting operations outside the Scrum Team is a false economy as the cost of waiting for support, or the overhead of communications, will reduce the agility of the team.
- **Does it make sense for these specialist skills to be learned by more people?** Agility requires generalists and by supporting the transfer of skills to Scrum Team members, operations becomes less of a bottleneck and more a part of the team. It also ensures long term flexibility as more people have more skills.
- **Is it important for legal reasons to have a separation of ownership?** This might not require a separate department, but increased transparency might be people outside of a Scrum Team have a view of the work. In the Open Source community, a similar need has resulted in automated code review tools and mob programming models. This could apply to other legal transparency requirements.

Accountability and Responsibility

Traditional approaches to development and operations have created different areas of accountability and responsibility. The DevOps movement has focused on removing/changing this. However, without shared measures, objectives and a clear connection to the customer, it is impossible for this to be successful. It is therefore important to align teams and teams of teams around the customer and the outcomes they seek. The container of these outcomes is the product

which might map across several applications and systems. To be successful with this change in orientation, organizations need to have:

- **One shared Product Backlog.** Today most organizations have several backlogs with each team having their own work lists and priorities. To create shared accountability and responsibility, it is fundamental to have ONE shared, visible, Product Backlog. This backlog should be managed by one Product Owner who is accountable for ensuring that value is delivered. They likely will not do all the work, but they are accountable for ensuring that it gets Done and is prioritized in a manner that delivers the most value in the end.
- **A single, outcome-based dashboard that everyone works toward.** If you want everyone to walk to the same drum beat, it is important that everyone knows what that drum beat is. By building a shared dashboard, Scrum Teams and operations will work to the same transparent measures.
- **A Product Owner who is accountable for what is delivered to customers.** It is too easy for Product Ownership to be focused on delivering new stuff, but only lightly responsible for the operational systems, cost and quality. DevOps needs to be inclusive of the Product Owner and make them part of the solution.

Self-Organizing and Empowerment

Healthy, high-performing teams decide how they are organized, how they work, and what tools and processes they use. To do this, they need to be held accountable for the value and quality of results they deliver, not micro-managed on how to get those results. No one approach works in all situations. The people doing the work, however, are the ones best able to decide how to do it. One size does not fit all, and a key learning from the Agile community is: by prescribing a fixed "Agile" process for every situation, you end up with the opposite of agility. Instead, focus on an empirical approach that starts from idea and finishes with customers. That means, clear alignment around customers for the whole value stream.