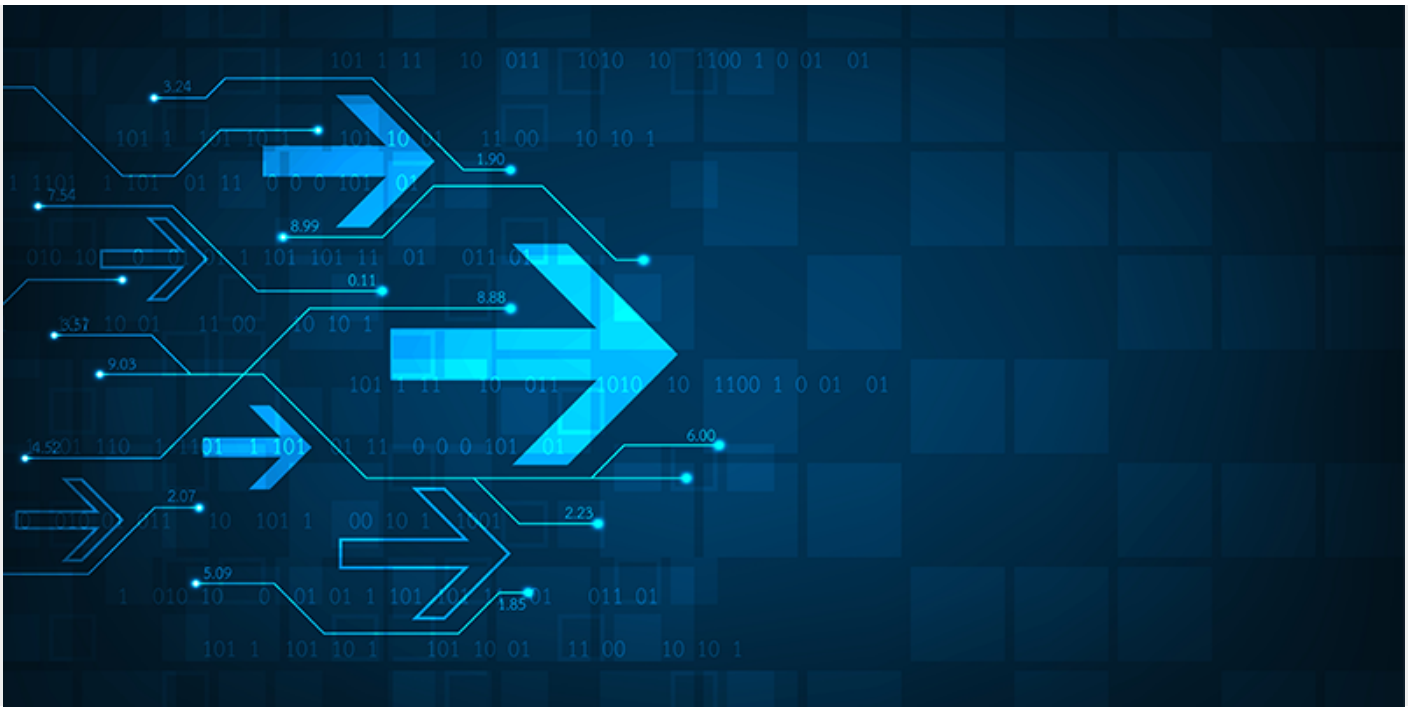
EVOLVING YOUR SOFTWARE DELIVERY SYSTEMS



Overview: The advantages of Agile and DevOps are significant and well understood. However, in many organizations these approaches have yet to be applied to one of the most vital domains, the software delivery ecosystem. In this post, we examine why it's so vital to apply Agile and DevOps to the software delivery system, and detail some of the key considerations for making the move successfully.

Whether you're in financial services, retail, manufacturing or just about any other industry, your software delivery systems are more critical to your business' success than ever. The ability to deliver innovative new software to market is integral to virtually every key business objective. Ultimately, whether an organization establishes a world-class delivery system can make the difference between success and failure.

The problem is that within many organizations, the approaches applied to managing the software delivery system itself haven't changed. These approaches remain very much like they were years, maybe even decades, ago. Teams are focused on projects. They employ waterfall-based approaches when making improvements. Ultimately, senior leadership views the software delivery system as a cost center, rather than as the strategic, value-generating vehicle it is.

Contrast this with how teams are managing the development of digital services they deliver to customers. In these areas, teams have been focused on developing and leveraging Agile and DevOps approaches. Their organizations have invested millions of dollars in leading technologies

and expended significant time and effort in establishing the expertise, workflows and capabilities required. They've assigned dedicated product resources and built scrum teams to implement new features and capabilities and reduce technical debt.

For an organization to be successful, it must approach its software delivery system the same way. By identifying the components of its software delivery value stream and defining processes that make them more cost-effective and responsive, an organization can maximize the delivery quality, velocity and efficiency of its software.

Defining the Software Delivery Value Stream

As with other "product" based software development efforts, the software delivery system is composed of several key steps and enabling technologies including:

- **Ideation.** Teams need to manage the new requests and ideas that arise. Tools like Confluence offer modern capabilities for capturing and tracking demand. These tools provide a centralized view so that ideas can efficiently be contributed, modified and properly vetted.
- **Backlog definition and prioritization.** Teams need to plan and define the work involved in executing on specific ideas and prioritize the various initiatives and supporting tasks that need to be done.
- **Software asset management.** Source code, scripts, configuration files and a number of other elements have to be managed, particularly in cases in which application software is being released on a more frequent basis and there are dependencies between assets. Management of these artifacts has to align with Agile principles to ensure speed and quality.
- **Editing and debugging.** Coding iterations need to leverage modern editors and integrated development environments (IDEs) with debuggers that allow for fast and intuitive fixes to modified code.
- **Build and testing.** Leveraging continuous integration and delivery capabilities is critical to delivery efficiency and quality. Automating the build process and integrating testing in that process will have a significant impact on the quality of output.
- **Deployment.** New capabilities and technologies need to be thoroughly tested and, when ready, automatically deployed to the next location in the delivery process.
- **Release management.** Enterprise applications delivered today typically are not platform specific. Application components spread across multiple platforms require modern release management practices that help speed delivery, enable potential back-out and recovery and ensure quality system delivery. Separation of duties, specific process enforcement and approvals and auditing are also drivers for release management tooling.
- **Monitoring.** As with any customer-facing service, it's vital to establish ongoing, continuous monitoring of the software delivery system to ensure it continues to perform optimally.

Creating a World-Class Software Delivery System

If organizations want to materially improve software delivery quality, velocity and efficiency, they should make managing and optimizing their delivery systems a strategic priority. This requires thoughtful investments in technology and building organizational structures and workflows that help make internal software delivery teams as fast, efficient and productive as possible.

In addition, teams need to employ the same Agile and DevOps approaches they're applying

elsewhere in their value streams to their software delivery system.

Following are the key strategies for applying Agile and DevOps concepts to the software delivery system:

- **Manage the software delivery system as a product.** In Agile organizations, resources dedicated to business value streams are responsible for the eventual delivery of products or services. In the same way, the software delivery system should be viewed as a value stream that produces the software required to support those applications or other deliverables. Like other products, the software delivery system should have a product owner and dedicated people who are responsible for the software delivery system's health, effectiveness, efficiency and continuous improvement.
- **Leverage Agile core principles.** Start by assessing current delivery systems. To kick off the transformation of the software delivery service, conduct a planning session and establish a product backlog. This backlog should define the incremental work needed to evolve the software delivery system. Use sprint planning sessions to define the minimum viable product (MVP) and sprints to build and test it. Additional sprint planning and execution should be used to prioritize the backlog and incrementally add capabilities and features to the software delivery system, enabling continuous improvement.
- **Develop a release strategy.** Annual release cycles don't cut it for business services, and they don't make sense for today's software delivery system either. Once an initial system has been deployed, it is vital to implement narrowly scoped changes in a fast, iterative fashion. It is important to determine how system releases will be rolled out to multiple user groups within the organization. There will be differences between user groups that need to be accounted for. Leverage quality assurance techniques, such as smoke tests. These techniques provide just as much value to the software delivery system as they do to other critical business systems.
- **Automate the delivery pipeline.** As teams continue to struggle with mounting demands and constrained resources, automation represents an increasingly urgent mandate. The more teams can leverage automation, the better equipped they'll be to realize the efficiency and productivity required.

By managing the software delivery system as a critical business value stream and leveraging these strategies, teams can deliver innovative software to market faster to meet business and customer expectations.

Conclusion

In many enterprises, teams have made significant investments in Agile and DevOps approaches—and seen massive benefits as a result. However, it's also very common to see that in these same organizations, these approaches have yet to be applied to the software delivery systems in place.

By applying Agile and DevOps methodologies to their software delivery systems, including a well-defined value stream, organizations will be able to continuously deliver higher quality applications to market faster. Additionally, a best-in-class software delivery system will also boost developer productivity, increase engagement, and attract new talent. Motivated and happy developers are passionate explorers for your organization and critical to your success.