Advancing DevOps Methodologies to Establish Best Practices
When it comes to transforming the modern software factory leveraging agile and DevOps methodologies, no challenge is more significant than affecting a wholesale change in culture.

Enter best practices.

While DevOps methodologies have been in use for a relatively short period of time, leading adopters have started sharing their experiences to create a template for success. Whether it’s understanding how to orient teams to optimize productivity, maintaining a close eye on software quality, ensuring that operation is emphasized or implementing measures to track efficiency, there’s a big opportunity to learn from existing expertise. Ignore these stories at your own risk.

In this eBook, leading DevOps experts at CA Technologies examine some of the critical elements required to drive DevOps adoption, and share observations that originated out of interactions with leading practitioners—from creating all new application delivery processes to monitoring application delivery lifecycle performance.
Collaboration: The Paramount Ingredient in the DevOps Recipe

It probably seems pedantic and redundant to keep driving home the message that DevOps cannot succeed without end-to-end collaboration, but time after time, real-world scenarios prove it’s true.

That collaboration is the most vital ingredient to DevOps success may already be covered in some of the proceeding chapters, if not underlined by almost all of them. However, thought leaders always revert to this theme when attempting to communicate that no other aspect of the application delivery lifecycle transformation is as critical to realizing the proposed benefits.

That’s why, much like in the time-honored tale by a similar name, CA Technologies DevOps expert Peter Waterhouse revisits this theme in his blog post “DevOps and Agile Lessons from the Story of Stone Soup.” Just as every stakeholder or team across the software factory may feel that some element of its changing workflow represents the most important piece of the puzzle, Waterhouse points out, it’s actually the joint effort spanning every domain that delivers the most-savory outcome for the larger process.

Countless stories exist where assertive management officials begin creating new initiatives, including workforce transformation, agile coaching and Scrum training, without understanding that independent of dedicated collaboration, none of these practices will produce the desired outcomes. “Our CIO has all her development teams itching to get cooking. Better still, folks are talking-the-talk with lively discussions around digital disruption, minimum viable products and failing fast,” Waterhouse notes. Yet, without the connectivity among teams across all of this work, the outcomes will still be fragmented.

The problem is that even when individual groups across the DevOps spectrum use these tactics and related tools to become more efficient, without end-to-end visibility and the ability to respond to issues by incorporating continuous feedback loops, people are still operating in silos. To truly enable speed and quality at scale, the collaboration mantra must be repeated, and then some, observes Waterhouse. Even when collaborative processes are in place and beginning to flourish, organizations can never take their eye off the game. Waterhouse posits:

“With collective involvement and the power of broader teamwork and contributions, great things can be developed that everyone will value. But just remember that whatever business we cook up from our digital menu, DevOps programs must strive for improvement whenever the taste gets bland.”

Whipping together the right blend of DevOps elements and constituencies is impossible without pervasive and constant collaboration.
The rise of agile and DevOps methodologies has created an entirely new set of methods useful in planning and in optimizing applications.

Over the years, many organizations have found themselves held captive by those applications considered too vital to take offline, yet too volatile or brittle to modify aggressively. But, thanks to the rise of agile and DevOps methodologies and tooling, we can address some of these applications—or at the very least improve them while driving replacements through to production.

Unlike those zany comic book ads of years past that offered do-it-yourself techniques for growing overnight from a pipsqueak into Charles Atlas, the availability of newer processes finally enables practical augmentation of troublesome apps, notes Waterhouse. In his related blog, “4 DevOps Strategies to Stop Your Application From Having Sand Kicked in its Face,” he highlights some of the steps that today’s organizations can employ to toughen up those apps that have proven difficult to address.
Waterhouse recommends related practices, including:

**Moving away from break/fix to proactive improvement.**

Because nearly all apps are becoming more mobile and cloud-centric, spend less time focusing on backlogged IT issues and more time on customer-driven problems.

**Avoiding the pitfall of shadow opps.**

Even though shift-left development techniques aim to address subsequent production flaws, ops must also evolve to keep pace and ensure stability.

**Leveraging ops as the ultimate source of quality control.**

Per the point above, operations teams must be empowered via critical feedback loops that directly influence ongoing code development and refinement.

**Transcending application resilience as a practice.**

Rather than attempting to play a constant game of catch-up related to apps performance, leveraging the DevOps-bred approaches of experimenting and failing fast to innovate and improve on the fly is where practices need to go.

If you’re buying into this approach, the general idea is that organizations need to be willing to truly let go of traditional applications operation and support mechanisms and invoke more adaptive, forward-looking methodologies. For instance, Waterhouse contends, organizations should focus more on practices like utilizing apps performance data to influence development, or integrating change management within the context of release automation.

“Highly distributed, scalable, cloud-native apps are very different, so our approach also has to change,” Waterhouse writes. “Therefore design and operational practices must move beyond break-fix, to ensuring inevitable failures have a minimal effect on customer experience and that each application release ‘learns’ from the experience—yes, actually gets stronger.”
Laying the Groundwork for Agile Operations

Software development is being transformed by agile and DevOps practices, but operations teams must also evolve to help ensure apps quality and performance.

One of the most widespread fallacies related to the emergence of agile and DevOps methodologies is that the importance of operations is being de-emphasized. The reality is that rather than being cast asunder by developers’ growing reach into core capabilities for which operations has long been responsible—such as standing up infrastructure—the rise of DevOps requires that ops teams similarly shift their focus to become even more influential arbiters of overall software quality.

In his related blog post “Why Agile Operations is Essential for DevOps,” Waterhouse outlines some of the specific skills that ops teams must embrace as they evolve practices to address emerging workflows. “It’s no stretch to state that even with all their new tools, aspects of operational excellence like maintainability, resilience and supportability are not top-of-mind for developers” he writes. “And even if there are concerns, one team’s approach to establishing these could be less rigorous than another’s.”

The point is not subtly stated; that being, even if developers are incorporating practices and tooling that help them cover some traditional ops responsibilities, that doesn’t mean they’ll transcend the need for support—such as that related to infrastructure management, performance monitoring and compliance, just to name a few.

Moreover, even as shift-left mentality pushes some tasks to the dev side, ops teams are still needed to craft capabilities that others will use “... to build operational excellence into the fabric of everything developed,” notes Waterhouse. For instance, as developers do more of the grunt work and use various forms of automation to stand up environments, they’ll still need operations teams to help augment that process and ensure that just the right manner of capabilities are available as developers unleash their latest code.

Perhaps most strategically, ops teams are increasingly being asked to begin monitoring the viability and performance of code builds before they ever reach production, a practice completely foreign within the realm of traditional waterfall practices. By feeding performance data over to devs even before new code is taken live, “... monitoring will come out from behind the ‘production curtain,’ providing teams with rich and realistic performance information for more reliable testing,” Waterhouse contends.

Here’s the reality: using DevOps, operations teams are actually becoming more strategic partners in the apps lifecycle than ever before. That’s one of the core tenets of the movement’s underlying promise, and one that shouldn’t be underestimated.
In Major League Baseball, specifically its vaunted Hall of Fame, the question lingers—will players from the juiced era of widespread steroid use ever gain recognition. Sports cheaters aside, similar practices useful in padding stats have begun to crop up in the realm of applications delivery as DevOps teams and program leaders seek to validate that their efforts are having a significant impact. However, like those ballplayers who decided not to take the quick route to stardom, DevOps practitioners should resist the urge to use “optimized” metrics that throw performance into a better light, and instead favor measurements that actually surface value.

In his blog post, “Measuring DevOps—Where Are You on the Cheat Scale?,” Waterhouse calls out these shady methods some teams use to play up their ability to make progress. For instance, even a semi-reliable stat such as Net Promoter Score (NPS)—widely referenced among transformational leaders—only tells part of the story. Even if DevOps initiatives are resulting in software that improves customer satisfaction (the outcome denoted by NPS scoring), there are a lot of additional factors that contribute to such a metric, making it troublesome to purely attribute such performance to software delivery.
As Waterhouse notes, more manipulative and common efforts to create overstated measures of DevOps success include:

- **Floating infrastructure**, such as when mobile app developers stand up unmonitored cloud services for testing to circumvent wait times, which won’t scale and may lead to subsequent performance issues.

- **Gaming bottlenecks** or delaying the introduction of time-intensive changes or updates until right before a release deadline, hoping that they get rushed or go unaddressed—something that’s problematic for obvious reasons.

- **Excluding architects** rather than working with colleagues who may introduce harsh realities around performance or scalability, and conveniently leaving them out to push through potential changes.

If any of these techniques sound familiar, you may want to take a hard look at how and what you’re doing in an attempt to create the appearance of increased velocity. The degree to which such practices are in use across your organization should tell you where you sit on the ROI metrics cheat index.

Despite the fact that some of these tactics could result in short-term gains, clearly, the long-term risk outweighs any benefits. Because really, does anybody want to become the Barry Bonds of the application economy?
Based on everything we discussed, you hopefully have a better understanding of how to proceed with advancing DevOps adoption across a wide range of applications delivery practices. From creating end-to-end workflows to measuring success, the emphasis clearly rests on the idea of maintaining consistent visibility to keep everybody involved in lockstep.

Now that you’ve taken huge steps to shift culture and adopt DevOps best practices, it’s a good time to review emerging patterns, ensure you don’t get hung up and begin creating a long-term roadmap for success. We’ll cover these concepts in our next series installment, “Envisioning the Future of Software and DevOps.”

For more insights into improving your process maturity, check out the newly published book, DevOps for Digital Leaders.