Overcoming common mistakes in DevOps, agile and app modernization

The four anti-patterns of digital transformation and what to do about them
Steeped as we are in corporate DevOps and agile transformation efforts, one of the common questions we get is, “What have other companies struggled with in their transformation attempts, and how can we learn from their mistakes?”

This has inspired a short list of what we see as the most problematic “anti-patterns” of transformation and modernization attempts, along with strategies for overcoming these tendencies. An anti-pattern is a commonly repeated pattern of action, process or structure that initially appears to be beneficial, but ultimately produces more bad consequences than beneficial results.

Each of the four anti-patterns in our list is understandably tempting. Many IT leaders cut their teeth in a waterfall, plan-driven and control-oriented world. It’s not their fault. An agile way of leading takes practice and learning, but it also requires a bit of unlearning.

While the terms “digital transformation” and “IT transformation” are used widely and at varying levels, in this paper we’re looking at efforts to radically improve business outcomes, speed and agility by leveraging approaches such as agile, DevOps, cloud and application modernization.

Let’s first walk through the basic reasons these four all-too-common anti-patterns result in a failure to meet even the basic expectations of such transformations, and are often even detrimental to business outcomes:

1. **Modernizing for modernization’s sake**

   We know the majority of organizations today are contemplating or doing some sort of IT modernization and/or digital transformation. Most are in some state of agile transformation and are looking to modernize their IT capabilities. A 2018 survey¹ by Statista showed that only 9 percent of the organizations queried had not started to adopt DevOps and had no plans to do so. Everyone else is transforming or getting ready to. A rapidly growing number of organizations are pursuing cloud, shifting to microservices, getting off mainframes or engaging in application modernization projects.

   There is nothing wrong with this, of course. Problems arise, however, when these efforts are pursued simply to modernize without specific outcomes or goals. DevOps, for example, is a means to an end, not an end state in itself. The same is true for cloud, application refactoring, mainframe migration and other application modernization endeavors. These are not easy undertakings and require a large commitment across the organization.

   Likewise, agile transformations should be done with an intent to get specific benefits for the organization. Without this, results may not be commensurate with the very large effort required. With both DevOps and agile, we’ve seen too many organizations stop short of realizing their expected gains because they did not aim for specific outcomes.
The desired outcomes should also be realistic and match what DevOps and agile are meant to do. Some leaders have misperceptions about what an agile transformation can accomplish. For example, we recently met with a team who believed agile would speed up code development and was trying to calculate the cost savings they could glean. But rather than trying to get a 5 percent productivity gain — which is not even the goal of agile in the first place — why not shoot for the larger gains agile affords?

The true benefits of agile are more profound and more strategic. Organizations that holistically adopt agile are able to respond rapidly to changes in the environment, including market, technology, competition and other forces. Another key benefit that dwarfs the benefits of development team efficiency is the ability to adjust plans quickly based on stakeholder feedback, to get the right capabilities implemented to best meet the desired outcomes.

Rather than meeting a set of established requirements, agile teams work to meet business outcomes. Instead of reducing the risk of not meeting a schedule, they reduce a bigger risk — that of not achieving desired results and outcomes by not delivering the right features.

As Mark Schwartz pointed out in his book, War and Peace and IT,² Microsoft found in a large study that, among requirements, “only 1/3 of ideas actually accomplished their intended objective; another 1/3 actually had the opposite result, and 1/3 didn’t have either effect.”

This is where agile shines — not in meeting documented requirements faster, but in implementing fewer of the wrong features and more of the right ones. The agile principle here is: Maximize the amount of work NOT done. Now, rather than a small IT development productivity gain, we’re talking about much larger and more effective productivity gains for the business overall. But if an organization just seeks generic IT modernization by “doing agile,” it may miss the greater gains and true power of being agile.

With a DevOps transformation effort, if the goal is simply to be doing DevOps, teams may reap some nice efficiencies or local optimization through the use of automation. But the biggest results come from targeting specific and measurable desired outcomes, and leveraging the pertinent DevOps practices and patterns to achieve those. Again, it’s not doing DevOps for DevOps’ sake, but instead meeting specific goals such as improved cycle time, measurable quality gains, and/or radical reductions in mean time to repair and change failure rates.

As the DevOps Research and Assessment (DORA)³ team proved in its industry-wide studies, companies that made the most use of DevOps practices, with relentless focus on specific measurable results, improved not just IT productivity, but also company-wide results. These companies were 1.53 times as likely to meet or exceed goals such as profitability, market share, units sold and operating efficiency. They had 50 percent higher growth in market cap over 3 years’ compared to companies that did not make this kind of deliberate and persistent use of DevOps practices. These organizations did not achieve the results by generically modernizing just because it was the latest and greatest IT trend.
Another very common digital transformation and modernization anti-pattern we’ve observed is where organizations attempt “big-bang” transformations. We know these add significant risk and rarely work. Regardless, such transformations are still often attempted, and as a result can set organizations back further than if they had never tried the transformation in the first place.

This is well-documented by many of today’s leading IT thought leaders. Jez Humble, Joanne Molesky and Barry O’Reilly in 2015 provided a good overview of the risks of this phenomenon in Lean Enterprise (a must-read for all technology and business executives and leaders). They show the amusing and all-to-familiar graph of organizations oscillating between “business as usual” and “change program” with a repeated sliding back to the previous state. Beyond a big-bang transformation’s lack of success, it has the larger risk of reducing the appetite of leaders for attempting more change.

The ultimate irony is when leaders use a big-bang, plan-driven, waterfall approach in efforts to move off waterfall (and go agile and adopt DevOps). The basis of agile is to be able to adjust plans and requirements based on knowledge gained from incremental implementation and feedback loops. This is a perfect fit for transformations, especially because of the inevitable issues, roadblocks and cultural inertia that can only be discovered through actual implementation attempts. An agile approach allows for quick and flexible adjustments when the roadblocks and constraints are discovered.
A misconception shared by many waterfall-minded leaders is that a strict and detailed rollout plan provides more control and less risk. Nothing could be further from the truth. In fact, the opposite is true. For anyone with doubts about this, read Mark Schwartz’s earlier book, A Seat at the Table, which provides a compelling (and humorous) view of risk, planning, requirements, governance and transformation — all from an agile leadership vs. waterfall-based perspective.

The other problem with big-bang transformations is that they tend to run long and yield results predominantly at the end of the transformation. Given the rapid pace of IT advances and changes, this often leads to having to start a new transformation as soon as the initial transformation has been completed.

The recommended pattern here is to do continuous transformation and use an agile approach. You get results as you go, and you have flexibility to continuously adjust as the IT landscape changes. Risk is reduced through transforming in small increments, making adjustments and removing impediments based on learning from the actions taken. Lower risk, more control, more predictability and more results! This is the antidote for the big-bang transformation anti-pattern.

One-size-fits-all adoption

This third anti-pattern, one-size-fits-all adoption, is most prominently seen with DevOps transformation efforts, but is also common with shifts to cloud, micro services, and several other transformation and modernization attempts.

Here’s how this anti-pattern plays out for DevOps adoption: An organization has some local successes with a couple of teams but struggles to scale adoption broadly. Using the timeworn management approach, it mandates “adoption” and creates a status-tracking mechanism measuring how every team adopts the tools and becomes “compliant” or “converted” to DevOps.

The problems with this approach are multifaceted. First, it rarely yields results. Second, DevOps is by definition a continuously improving endeavor, not something you finish, become compliant with or convert to. Third, and the part most often missed, is that DevOps includes a myriad of principles, practices, patterns and tools that can and should be applied selectively and in order of priority, depending on the specific needs, goals, current state and challenges of each specific app or value stream. Sure, there are a few common tools and some common organizational DevOps patterns that may make sense to apply universally, but they are the exception, not the rule.

The trick is to NOT assume that all value streams in one organization have the same current state, goals, needs and challenges, and NOT to treat DevOps adoption as something that all teams must obediently comply with and “check the box” — unless of course the organization just wants to claim it has completed a DevOps or agile transformation but isn’t really interested in the results.
There is also a very important human factor with avoiding the one-size-fits-all anti-pattern. Gary Gruver, an early DevOps transformation pioneer from HP, presented a common scenario: You mandate skeptical team managers to implement DevOps, claiming it will dramatically improve speed and quality. They reluctantly comply, get limited results and say, “See, it didn’t really work.”

On the other hand, if you hold the team managers accountable for specific speed and quality objectives, provide them a framework of agile and DevOps principles, practices and patterns (along with coaching and training support) and let them determine the best way to achieve those results, teams will own the results and accomplish objectives much more effectively than any mandated checklist.

The one mandate we recommend that executives should make is for the team to adopt a disciplined, continuous improvement and transformation process (aka DevOps Kaizen). By setting the business objectives and driving the continuous improvement process, executives will see much better transformation results than by mandating a one-size-fits-all checklist.

Technology and tools-only focus

This is the grandfather of transformation anti-patterns. Intellectually, nearly everyone understands this — that tools are responsible for 20 percent of any success realized, and 80 percent of success is about how teams work (culture, leadership, practices, principles and mindset). But when it comes to actual prioritizing, emphasis and steps taken during transformation efforts, the majority of organizations expect transformations to DevOps, agile, cloud, etc., to be successful by simply rolling out tools, methodologies and life-cycle processes.

The tools are the easiest part of the equation, while the culture change is more challenging and more difficult to measure. Focusing only on 20 percent of the challenge (the tools) will, at best, get you only 20 percent of the benefits.

This critical 80/20 rule is emphasized in countless DevOps/agile leadership workshops and seminars about critical success factors in transformation, and heads always nod in agreement and understanding. Then the questions start, and the majority are about tools.

We also see it with companies looking for help and advice on how to scale DevOps adoption. They have experienced pockets of success with a few small unicorn teams but are struggling with scaling beyond that. They explain how they have provided and mandated a common toolchain. But they are not seeing widespread adoption, nor are they getting expected results. Tools alone are not the answer.

The critical success factors for DevOps mostly center on how teams work. This includes continually adopting rapid feedback loops, continuously optimizing end-to-end flow, continual improvement through hypotheses and experimentation, ownership of outcomes, collaboration, eliminating silos, automation, shifting left, empowering teams, etc. Tools play an important role, but the tools are not the solution. DevOps is not a set of tools or the “onboarding” of tools. Not even close.
A classic example is Jez Humble’s famous three questions, where he asks audience members to raise their hands if they are “doing continuous integration,” which is one of many DevOps practices. All those using a continuous integration tool, such as Jenkins, and doing automated builds and automated testing typically raise their hands.

Humble proceeds to have people keep their hands up only if, “1) all the engineers [are] pushing their code into trunk / master (not feature branches) on a daily basis, 2) every commit triggers a run of unit tests, and 3) when the build is broken, it is typically fixed within 10 minutes?” The hands successively drop and, by the end, there are very few really doing continuous integration and hence really getting expected results. But the leadership team has rolled out the tools, and the teams have “adopted” them.

A similar anti-pattern exists with agile. Teams roll out agile by adopting not only the tool, but also all the important ceremonies, such as daily stand-ups and sprint reviews, for example. These are essential initial steps, but they don’t mean these organizations are reaping sweeping benefits. Agility is about much more than methodology.

As described in our first anti-pattern, many of these teams “do agile” but are not “being agile.” They may go through the motions, using the tools and the methodology, but without truly incorporating the agile principles, continuously improving, and adopting agile end-to-end across functions (especially across leadership, governance and business), results tend to be minimal. Local optimization when adopting agile is a very common variation of this anti-pattern, sometime referred to as water-scrum-fall.

**Antidote to the leading transformation anti-patterns**

Given these four primary transformation anti-patterns, what is the antidote? What are the best ways to prevent and avoid these traps?

The first step is to understand and recognize these patterns when they happen. It seems obvious, but waterfall ways of thinking are ingrained, and organizations are often unaware of the underlying mindset, culture and paradigm by which they are operating. In addition, we need to replace these with proven principles, disciplined implementation methods and known successful transformation patterns.

Recommendations include:

- Think of transformation as a continuous process that never ends rather than one big transformation or even a set of large projects.

- Use tried and true agile principles and practices for the transformation itself, just as a team would for agile product development or maintenance, such as:
  - Prioritized epic backlogs
  - Hypothesis and action-driven rather than plan-driven
  - A focus on incremental results of the transformation
  - Doing small improvements followed by quick assessment and adjustments
• Use a data-driven approach, whereby transformation efforts are applied to value streams based on their specific outcome goals and challenges. Adopt value stream mapping (the right way).

• Align leadership at all levels to support, encourage and model a culture of continuous improvement where teams are empowered and accountable to measurable value stream goals. This involves:
  - Sponsoring and tracking goals and kaizen iterations
  - Addressing organization-wide impediments
  - Empowering teams
  - Helping the organization fundamentally “get good at getting better”

• Create an enablement team of coaches, rather than a heavy governance and/or compliance team, such as a traditional Program Management Office (PMO).

• Partner with a service provider that applies these principles and approaches in its consulting, enablement and delivery offerings. This partnership can help an organization become self-sufficient in its continuous transformation and modernization efforts.

There are two core threads common to all these principles and patterns that are critical success factors:

• Leadership and culture

• A system or framework for continuous transformation

First, leadership culture change is paramount. This culture shift requires multiple ingredients (too many to embellish upon in this paper), but these include a burning platform, a clear vision, and continuous development of executives, leaders and practitioners. A good transformation consultant is highly recommended to help with this. As DORA11 has demonstrated, culture can also be influenced and changed through consistent application of practices, such as continuous delivery. In other words, the adoption of technical practices from the bottom up can actually shift the broader organizational culture.

The development of leaders and practitioners cannot be underestimated. DevOps and agile practices and tools have minimal impact without all levels of the organization understanding and embracing a body of core philosophies that have revolutionized both IT and business. These include, but are not limited to, the agile manifesto, systems thinking, and DevOps lean product development, which entail elements such as value stream management, three ways of DevOps, kaizen, minimally viable products, and lean-agile leadership.

This may sound like a lot, but these principles are the underpinnings of the absolute best way of developing and managing systems today. They are proven scientific principles that continue to evolve and improve. They are not fads or trends, and they should be considered mandatory elements of the continuous learning and development curricula by leaders and practitioners alike.
As part of successful continuous transformations, leaders must understand, support, and apply lean, DevOps and agile principles and practices.

The second core thread is a system or framework for continuous transformation. Think of this as a continuous transformation factory that builds on all the principles listed above. We have developed such a framework to help clients scale, prioritize and sustain this approach across their enterprises for continued results with agility.

This is the ultimate antidote to the typical transformation shortcomings due to the all-too-common anti-patterns described in this paper.

About the author

As a Principal for DevOps Transformation, John works with external clients and internal DXC organizations to help consult, advise, and drive DevOps transformations from the executive to the team levels. John has been an IT leader for over 25 years with a variety of R&D Functional Manager and IT Director roles from leading large development organizations in multiple domain areas to running enterprise architecture, data architecture, operational excellence, testing, high availability, and numerous others. As a recognized thought leader and change driver, John is applying his leadership skills, engineering experience and passion to help drive enterprise-wide improvements and continuous transformations.

Learn more at www.dxc.technology/application_services
References:

2. Mark Schwartz, War and Peace and IT (Portland, OR: IT Revolution, 2019)
9. Mark Schwartz, A Seat at the Table (Portland, OR: IT Revolution, 2017)
10. Martin Fowler, Continuous Integration Certification, https://martinfowler.com/bliki/ContinuousIntegrationCertification.html