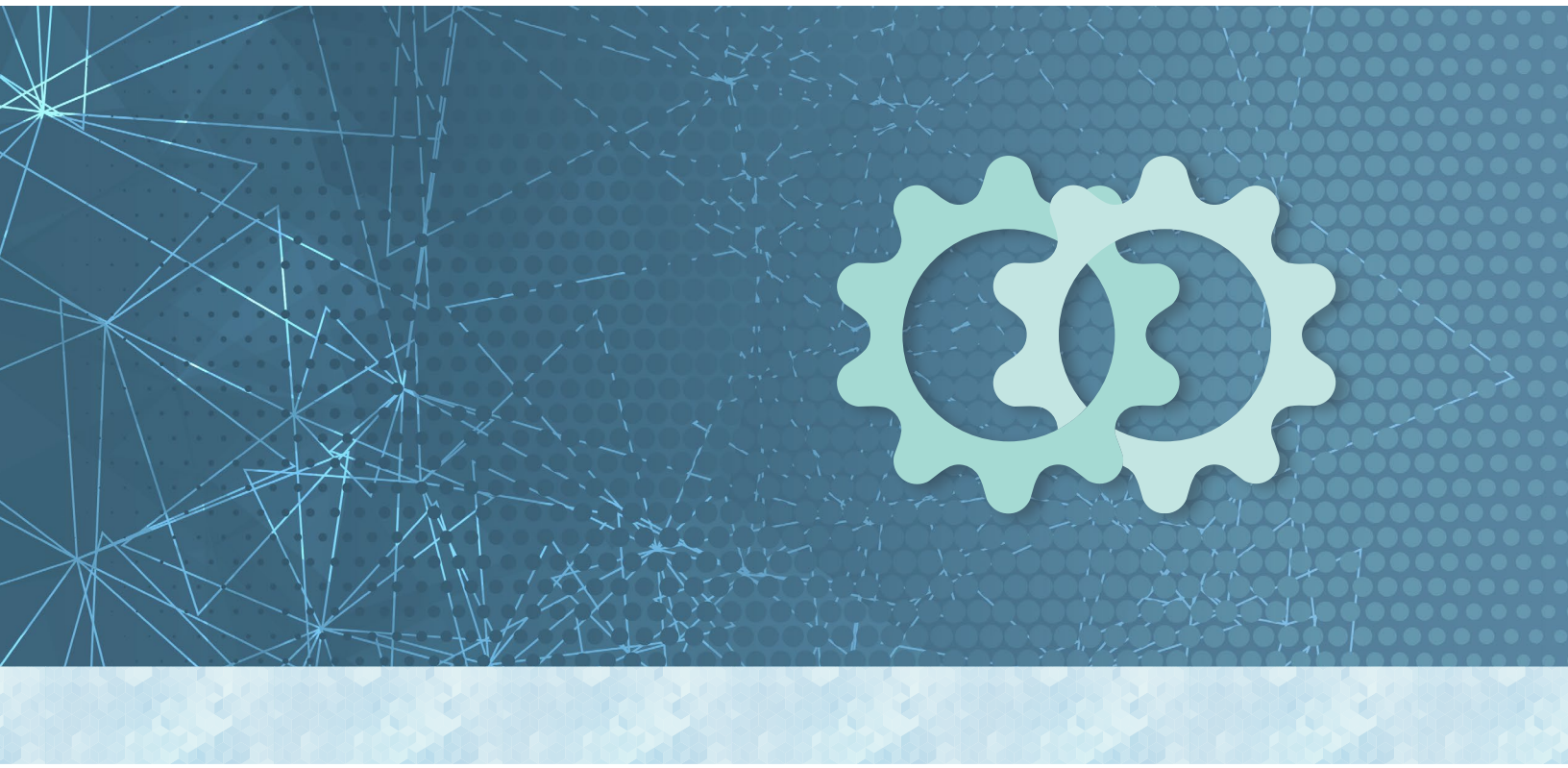

Effective SDLC: The Lifeblood of a Modern Organization

sdlc partners

Performance Enabled.

Contents

Introduction

The **Software Development Lifecycle (SDLC)** is becoming increasingly complex alongside users demanding increasing functionality at an ever-accelerating pace.

Out of the six standard SDLC models (Figure 1), our organization recommends a DevOps approach for most clients. We look at DevOps as creating a modern software factory for your technology teams to deliver extraordinary products to your customers.

DevOps emerged from two industry developments, using Agile and Lean practices and a shift towards more tightly aligned development and operations staff throughout the SDLC stages.

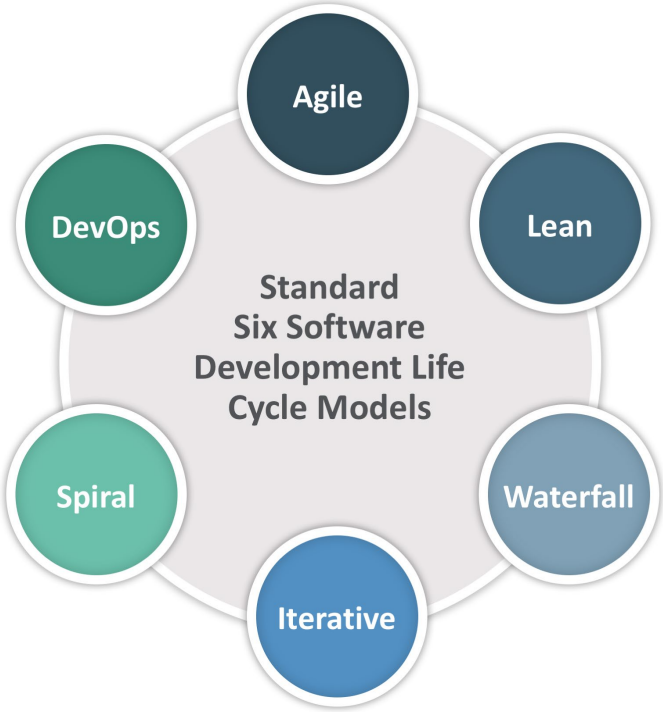

Figure 1: Standard Six SDLC Models

As described by Amazon Web Services, "DevOps is the combination of cultural philosophies, practices, and tools that increase an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes."

We have seen the value of using the best of Lean and Agile in a DevOps approach, which helps streamline software development while it addresses many of the typical problems faced by teams:

- Lofty list of improvements that just stew on the back-burner
- Speed-to-market is far too slow
- Output quality isn't meeting customer expectations
- Standards aren't clear, and execution patterns and practices are inconsistent
- Software development teams are mired in poor communication and their efforts are trapped in silos
- Collaboration between business requirements management and engineering teams is weak
- Software rework is costing you time and budget
- Operational metrics are unpredictable

With these challenges in mind, DevOps is much more than a system; it is a team-organization structure and a working philosophy for thinking and acting; a new cultural norm. If you view DevOps as creating a modern software factory for your technology teams to deliver extraordinary products to your customers, we hope you will quickly see how this context starts to pull every element together.

This whitepaper highlights the key challenges and reasons that a DevOps approach to SDLC is needed, as well as provides our best thinking to guide your decision process and next steps to improve your application lifecycle and software deliverables.

Lastly, you'll find a client case study that drives home the value of reimagining and enhancing your SDLC for better products, prouder employees, and happier customers.

Top Three DLC Challenges

Your development life cycle can quickly become derailed and, yet, it can be challenging to find the culprit. Evidence of a derailment includes failed or late releases, costly rework, and executives stepping into the fray. Together, these create a spiraling effect, lowering executive trust, and crushing the team's enthusiasm and passion.

There are three challenges that we see with almost every client who requests our assistance with their SDLC. Resolving these can mean better process control, delivering higher quality products faster, and a reinvigorated team who has more commitment and drive for creating world-class output.

Challenge #1:
Human and organizational habits are eroding digital product quality and competitiveness.

A lack of process, control, and tracking key performance indicators (KPI) leads to staff feeling like they have to reinvent the system every cycle. A proven, reliable process gives the team the confidence that their work is achieving a desirable end-goal and follows a logical progression.

The ability to deliver sustained quality and value in the market relies on processes that enable teams to work in harmony and reach the common goal. That process should be documented in a form that seasoned veterans and new team members can reference for consistency of execution.

Challenge #2:
Software development processes vary and problem processes undermine standardization.

Somewhat related to challenge number one, there are variables within processes that keep the SDLC from realizing the right features are built for the right reason in a timely and quality manner.

Teams want to standardize quality processes, and that begins with a clear view of the current state of one's SDLC and output quality.

Measuring your delivery performance against market expectations is critical, as well as measuring your execution performance against your standards and expectations can help you standardize what works well.

Create a clear definition of your process and desired standards based on a disciplined culture of continuous improvement fundamental to high-quality, engaging products and services.

Challenge #3:
Many SDLCs do not have adequate governance or systems for continuous improvement.

The highest value in a standardized, documented, and measured DLC is better governance, leading to uncovering issues and risks sooner. This helps you avoid the cost of botched releases and failed launches, along with the untoward effect of declining executive trust, decreasing team morale, and customer perception.

Even though everyone craves "faster...better...cheaper," in a mature and optimized SDLC, each step in the cycle should fuel continuous improvement because of adequate governance. A "build and deploy" process that is simplified, and automated, uncovers issues before expensive manual rework is necessary. Your ability to "shift left" leads towards a continuous flow that can make it easier to deliver a better product to the market ahead of the competition.

These stand out as the top three challenges to most SDLC processes and teams. There are indeed more, but these are the foundation for ensuring that you are releasing the best digital product possible and expected.

In the following sections, we will outline our DevOps process and why it is the most relevant and reliable way to manage your SDLC.

"With ever-growing customer demands and market competition, it has become paramount that we improve our delivery lifecycle and SDLC has ensured that we have done that."

Casi Johnson,
Chief Operations Officer/
Innovations Leader

M3 Accounting + Analytics

Journey to Enterprise DevOps

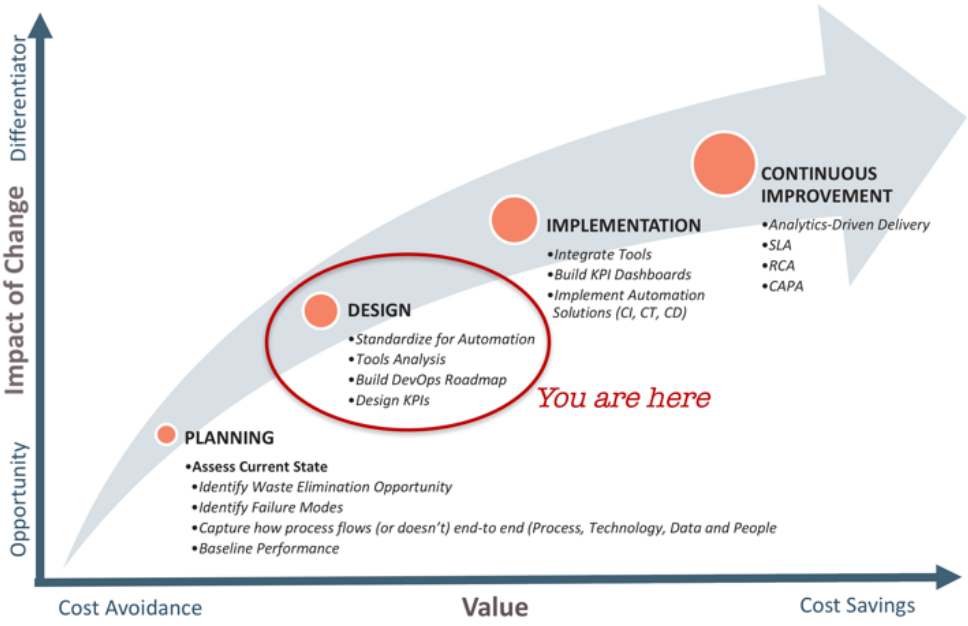

Figure 2: Journey to Enterprise DevOps

Disrupting Traditional SDLC with a DevOps Approach

Having a clear view of the challenges your SDLC encounters, and an understanding of the current and desired state of your development system, are powerful first steps.

Ultimately, a robust DevOps system gives you the ability to deliver what the market demands through a standardized, automated, and well-governed system.

DevOps focuses on using practices that emphasize collaboration and communication that enable building, testing, and releasing applications rapidly, frequently, and reliably.

Several steps are vital to the culture change that is central to successful DevOps adoption outlined here.

Why Commit to DevOps?

The DevOps approach brings together four teams that are traditionally disparate in larger organizations – operations, software development, quality assurance, and testing.

While each area has their role to play in a typical SDLC, at times, instead of collaborating to drive team success, the focus can turn to the individual deliverable measured against 'when' it is delivered to the next phase. Often this 'set and forget' mentality doesn't realize the deliverable through to the required definition of 'done.'

Driven by increasing pressures to improve customer experience and bottom line results in an increasingly competitive landscape, organizations are looking for the types of benefits that DevOps delivers, which leads to continuous delivery.

Embracing DevOps brings each team together, systematically, so that every group is more successful by everyone achieving the goal together.

DevOps goals should drive a production-first mindset that ensures:

- a standardized set of environments
- decreasing the failure rate of new releases
- shortening lead-time between versions
- a faster mean time to recover between releases

Ideally, a DevOps approach creates a situation where the organization releases more valuable software faster (or even on a continual rollout basis). A faster, healthier cycle-time equals more revenue, market share, and competitive advantage, as well as more satisfied and loyal customers and employees.

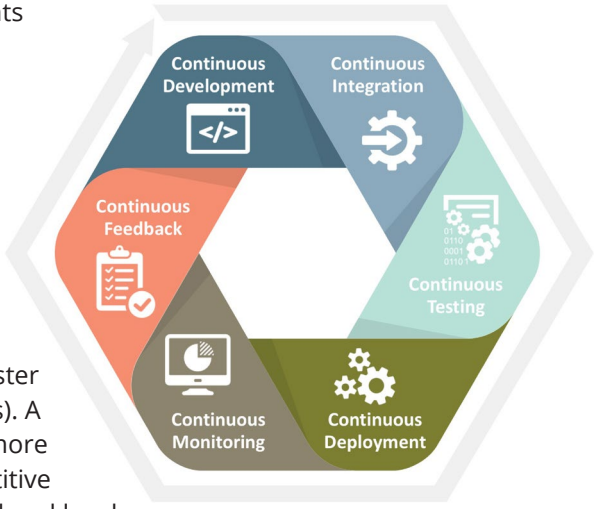

Figure 3 Six Phases of DevOps Model

Our Approach to DevOps

We take the best of Agile, Lean thinking, and human-focused design, and incorporate them into an approach to evolve teams into a cohesive DevOps shop.

This DevOps approach is built upon four pillars:

1. **Ensure that the program has full support from an organizational change management perspective**

The cultural change must start at the top and key executive or DevOps representation should be frequent guests during milestone meetings to provide updates on status, what's coming next and, most importantly, why it matters.

Some of the typical points of failure in implementing DevOps transformation successfully include:

- Inadequate understanding that DevOps is more than a "developer thing"
- Failing to define baseline and target metrics to measure progress on objective data points
- Bypassing a pilot project to drive out risks that could reveal knowledge before a broader rollout
- Not ensuring that everything (from tooling to process and cultural changes) is in place before rollout
- Allowing "backsliding" or mixed modes of implementation

2. Standardizing What Works

Aligning and reducing cycle times through gathering systems and establishing standards end-to-end. This can include:

- Defining common endpoints and systems interfaces
- Logging and transaction instrumentation frameworks that can be seamlessly incorporated into the process
- Coding and development standards that are readily available and commonly understood before broad adoption of DevOps
- Solidifying data governance and usage policies to combat sprawl and quality issues
- Maturing the API and microservices strategy so everyone can build loosely coupled, highly cohesive systems
- Establishing a baseline for business and technical architecture so there is always an active master plan to provide context for development efforts

3. Automate or Die

Automating "build and deployment" processes via continuous integration is a must-have: without it, you won't be carrying out DevOps successfully. Why?

- Having an automated, end-to-end delivery pipeline ensures that the correct process is being followed because it's codified and doesn't require every single developer to know every single step in the process. They can't make a process mistake
- Continuous improvement becomes actionable because the newly-discovered process and technical advances can be built into this delivery engine
- Software quality becomes more than tested code as controls, instrumentation, and standards monitoring and inclusion are being built into the tooling, processes, and methods
- Risk mitigation is more successful. From scanning the code for bad libraries, preventing merge issues, or stopping promotion issues, ordered steps and controls act as quality gates without slowing things down or missing processes

4. Embracing the Retrospective between Iterations (aka continuous Improvement)

Leverage Process Failure Mode and Effects Analysis (PFMEA) and Corrective Action/ Preventative Action (CAPA) as tools to drive execution control discipline and avoid recurring issues. Continuous improvement goes beyond "lip service" to the very action that enables you to delight customers. How?

- Teams can directly improve their software factory capabilities rather than create a learning loop or project post mortem that produces little value and boring PowerPoint slides
- Process changes, controls, and checks can be built into the pipeline

- Key testing misses can be remediated and solutions can be made available to every developer
- You will wake up one day having a successful, productive software factory that delights customers

Ready to Adopt DevOps?

As Stephen Covey is quoted, "Seek first to understand and then to be understood."

Before considering the advanced techniques involved in being an effective DevOps machine, you need to understand your current state environment and competency.

In Lean thinking, this process of understanding is called Value Stream Mapping. It helps you visualize how the process, technology/tooling, data/intelligence, and people flow (or not flow) to deliver value to your customer.

This holistic view of your product allows the right focus on what to do first based on the quantitative and qualitative impact a particular gap or failure incurs in your current state. Then, it's down to applying the right techniques to solve the problems by priority.

Start Disrupting Now

Disrupting the status quo is never easy. However, unmotivated, and fractured software teams can cause enough pain (mental and monetary) that makes change more appealing.

Creating a culture that's supported by streamlined and reliable DevOps can mean the difference between being number five in the market and number one. Freeing up your most valuable resource to innovate versus renovate is vital.

Five Must-Haves to Ensure DevOps Success in Your Organization

The following five areas are critical to ensuring DevOps gets off the ground and delivers sustainable results. These can help organizations get over the cultural, communication, and intellectual humps to drive DevOps success.

1. Design a Compelling (and specific) Future State

If the combined DevOps team can't see the value in pursuing common goals, nor the contribution that the "other" side makes to the whole, you'll have a tough time getting anywhere.

However, if the group bonds over designing what their desired future state could look like, they can have clues as to what success should feel like and know when this new DevOps way is working.

2. Capture an Honest Current State

Frankly, it's too easy to pretend everything is going fine or to be so mired in disharmony as separate development or operations teams that you can't see the issues.

Related to item number four -- it's essential to create a space and an opportunity that both teams can honestly uncover blind spots and hidden (yet obvious) issues.

DevOps transformation offers a fresh opportunity to encourage intellectual curiosity like never before. Tap into the strengths of the team and step out of the arena by pulling folks together to solve problems in collaboration.

3. Establish Process-focused Methods for Bringing Teams Together

Process is your secret weapon for collaboration. By having a fruitful and encouraging process to guide your DevOps creation and launch, you take a lot of the pressure off of the idea of two becoming one. Value Stream Mapping is one of the most potent ways to achieve a realistic view.

4. Attack the Problem – Not the People

It can't be reiterated enough. Your people are (usually) not the problem. And, by focusing on the issues and problems to be solved (supported by a dynamic process and facilitation), you can stay clear of digging up sore spots and old wounds and keep people on one side and the problem as the enemy.

5. Empower Staff to Take Leadership

This is the toughest to say and for organizations to achieve. Leaders must kick off the process and framework, but then get out of the way.

Make this experience about them and what they build for you and your customers. Don't make this about what you want them to do.

Establish clear lines of communication and feedback, but you will get your money's worth of time and energy if you give your team space and tools to design the solutions that they, then, will execute. There's nothing quite like the drive teams have to prove their ideas do, indeed, work.

A Practical Guide to Effective DevOps

Our DevOps model includes six areas to help you achieve process standardization, tool identification, and performance indicators necessary to drive effective execution control.

These phases will enable you to map your current state, design a future state, create a prioritized roadmap and, most importantly, dive into focused, measured implementation and feedback.

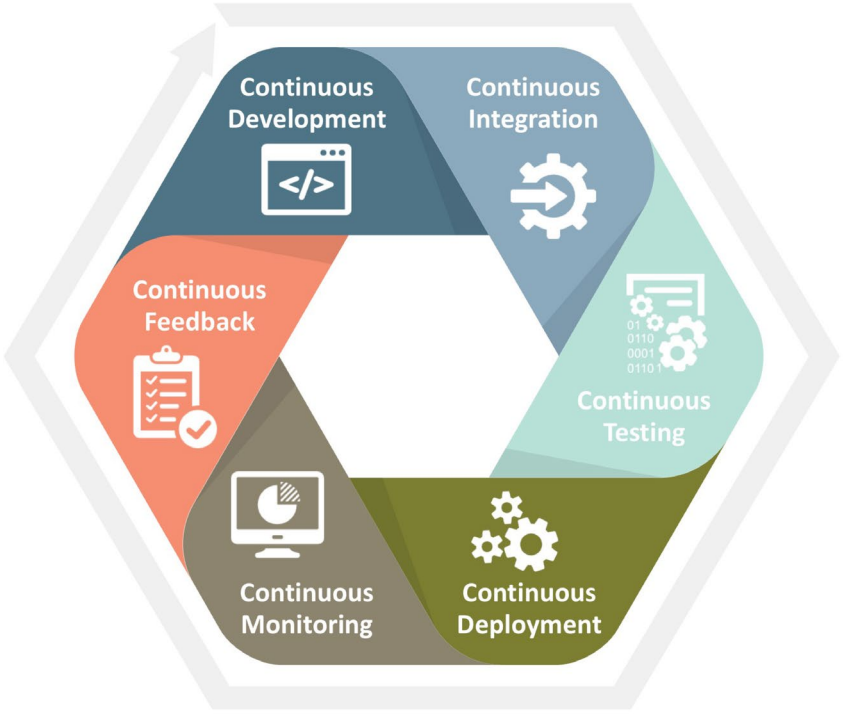

Figure 4: Six Phases of DevOps Model

Phase One: Continuous Development

Before we move to specific technology solutions to automate how we integrate and deploy code to your environments, first, ensure that your current software development methods are sound.

Assuming that you mapped out your Product or Software Development Lifecycle effectively during the planning phase, you should see how well your releases are planned, how effective your iteration commitments are, and what your baselines are for your key performance indicators. You may have even defined a more straightforward approach (future state) to eliminate the waste identified during the exercise. If you didn't, you would want to do that since you'll need a model that benefits from continuous services.

When it comes to the process for successful continuous delivery, ensure that you're following the appropriate Agile techniques for your product release and iteration plans. Driven by your product roadmap, you should have a healthy, unrefined (proposed) and refined (accepted), backlog of requirements for implementation. Your requirements are the backbone of continuous delivery. Without them, it's tough to project what work you need to plan and continuous delivery isn't feasible if the current iteration clouds your view.

Once requirements are stable, it's time for execution control, and that entails a well-configured Application Lifecycle Management (ALM) tool like Microsoft's Team Foundation Server or Hewlett-Packard's Quality Center. These leverage the best Agile or Capability Maturity Model Integration (CMMI) template to organize the team's tasks. Consider this your source of truth for implementation and measurement across the DevOps model. Used properly, this is your most powerful ally. Misused, however, it's another nuisance tool that no one will understand how to use.

Last, but probably the essential step, institute real-time performance metrics deployed as visual management and delivered through your ALM dashboards, showing team and individual contributions made to iterations. This is essential to effective commitment delivery and will help you identify impediments affecting the team's burn-down rate (remaining tasks to do versus time) and their velocity (speed of progress). Installing visual management next to every Agile crew is an effective way to celebrate success, promote teamwork, and can act as a useful tool to drive daily stand-up activities.

Phase Two: Continuous Integration (CI)

Without the ability to continuously integrate developed code, there is no continuous delivery. If the code is the blood of the development engine, CI is the intravenous system, providing a vessel to package, validate, and deploy code to the staging environments for action.

Selecting the right CI server, to centralize the build environment, is fundamental to success. There are many options — some offering configuration data in accessible files where job creation can be easily scripted. Whether you're working with virtual machines or containers, like Docker, we recommend Jenkins paired with tools like Puppet and Chef, which provide more advanced support to stand up the required server instances.

Instrumentation at the job level is key here, ensuring the appropriate notifications are set up to roll out the next phase of work and avoid expensive wait times. We want to prevent the QA team waiting on the next build while the deployment engineer wraps up for the day. Failures like this can lead to unpleasant after-hours troubleshooting for your staff and erode productivity, resulting in missed commitments and avoidable backlog burn-up. If your team is spread across different time zones, this is critical to nail down.

Phase Three: Continuous Testing

Now that you have your process stabilized and your tools enabling execution control, it's time to tackle testing.

You may already have some automation coverage, but the traditional approach taken to testing won't likely keep pace with your new development cycle times. This leads to delayed releases or worse — gaps in coverage and costly post-release defects.

When implementing a continuous testing solution, safeguard that you have the right balance of coverage and speed to protect the product as it moves between deployment gates. This requires taking your automation beyond the QA execution layer.

Before considering full-scale test automation, confirm that your environments are streamlined across the delivery architecture. Having a test suite that runs in QA and breaks in production won't help. Discern that the code being deployed will work across all staging environments through to production.

Once your environment is set, look at existing manual test cases to determine if the right end-to-end coverage is in place. Think beyond regression. If you can automate 95 percent of the value-add test scripts, you can spend the workforce in exploratory testing and analysis, which provides better insights into release viability. And, since automation was never intended to replace your people, it can augment what they do, magnifying efficiency and delivering impressive results. But, get the standardization wrong, and you'll see a higher count of the same problems faced before automation.

For tooling, check that automation runs are integrated into the ALM analytics engine. This keeps your "source of truth" clean and provides a consolidated approach to analytics across the lifecycle.

Finally, there are some powerful tools disrupting the industry like Kobiton. It provides a low-code automation engine, as well as multi-modality interfaces for testing in the cloud. Benefits from these tools go beyond cycle time improvements and positively impact traditional capital expenditure where you can cut your consumer electronics spend each quarter. It's worth the time to investigate.

Phase Four: Continuous Deployment

At this point, you have a sound lifecycle, you are seamlessly building committed code into releases across staging environments, and you can obtain quick feedback on performance and accuracy with a continuous testing solution. But, how does continuous deployment help?

Simply put, this is 'THE value part' – where all of your hard work designing, building, integrating, and validating code across staging environments is delivered to the customer. Inventory is the single worst waste type in engineering.

Leveraging the same technology used to build the CI solution, your live application in production is updated as well.

And, like every other phase of the DevOps model, instrumentation plays an important role. For continuous deployment, that role is critical since your customer can now see how the new features perform.

This is the point where each piece of the DevOps model plays in concert to validate what you released to the customer. Any failure discovered here should suitably abort the deployment and trigger a real-time intervention to determine the source of failure.

Of course, if you are maintaining a continuous testing model in parallel to the product feature set, and environments are set appropriately, this should never fail. If it does, the root cause analysis should discover a failure upstream in the process. Most likely, the environments have not been adequately maintained, or automation is no longer in sync with the feature set.

Phase Five: Continuous Monitoring

If we have applied the right systems thinking to build out the DevOps components up to this point, it's time to reap the rewards.

Even though continuous monitoring is near the end of our DevOps model, ongoing monitoring is vital as you build your key performance indicators for each area of the model. As the saying goes, "If it's not measured, it's not managed."

Most of the delivery intelligence should be housed in the ALM, plus you will have additional insights to gather from system logs tied to security and performance. And, because most ALM tools focus on the execution layer of delivery and less on the continuous improvement side, it's important to periodically review insights and create corrective, preventative actions that are being assigned to functional leaders. This may require a quality management solution.

Phase Six: Continuous Feedback

As you refine and add to the product backlog, reach out frequently and openly to customers about their experiences. Plus, gain insights from the production layer automation, regarding any operational exceptions that need to be addressed. These inputs, together, can help balance needs and wants as you plan future releases.

An application delivered faster with higher quality does not guarantee that you will achieve the desired business outcomes — how the customer benefits from those features determine value. To stay relevant in the ever-evolving solution space, you'll want real-time insights into their needs as their experience with your platform evolves.

Traditionally, organizations have sent questionnaires or hosted customer forum groups to collect such insights, but there are integrated ways to do this today — living within your product. We are partial to Qualtrics, but there are many user experience tools to choose from, ensuring there is balance in your product design thinking. Plus, with the flexibility you have created through this process, the team should have room for those projects that tend to be ignored – like innovation.

SDLC Client Case Study

Using Lean Thinking to Revolutionize Hotel SaaS

ABSTRACT:

The hospitality industry's leading provider of hotel accounting and analytics software wanted to accelerate new value and modernize functionality to deliver novel levels of value, usability, and quality.

The SDLC Partners team was able to help this client gain a clear picture of the patterns and practices necessary to reliably deliver value to their hotelier customers nationwide.

Collaboratively, we were able to create a customized development lifecycle built on industry-standard techniques. This gave them the flexibility and power to create a more aligned software product, faster.

THE CHALLENGE:

This first-time client had a lofty list of improvements they wanted to put into the hands of end users, which meant they needed to take a re-reinvigorated approach to their development process and systems.

Although they had used other outsourced help and consulting, they wanted to enhance their speed-to-market alongside output quality to align with what their customers wanted.

In addition to product modernization goals, they also had the desire to improve the following:

1. improved communication and reduction of silos;
2. increased collaboration between business requirements management and engineering teams, and
3. reduced rework.

For any organization, especially one built on a SaaS model creating technology for a niche industry, it's critical to develop what you promised and ensure old and new customers find increasing value from the web-based, subscription service.

THE SOLUTION:

The SDLC Partners' engagement philosophy was of particular interest to this client. We didn't come in and tell them what they ought to do. They had gone that route with other firms and it took them further away from their goal.

Instead, our model involves looking at the plethora of best practice models available and suggesting an approach that met the client's specific challenges.

In this situation, the Value Stream Mapping tool, derived from Lean thinking, seemed like the best way to bring the team together and uncover the gaps in value and

"Our employee survey results give me tangible evidence that our development improvements have increased satisfaction by 39% over last year for executing well. Now, we manage our development process efficiently and with excellence."

Casi Johnson,
Chief Operations Officer/
Innovations Leader

M3 Accounting + Analytics

process. Essentially, we met them where they were and helped their team create a custom development lifecycle that addressed their identified needs and those of their expanding, global customer base.

Together, we helped them identify five decisions that they believed were paramount to changing direction and making up for lost time, including:

1. **Fostering a more collaborative requirements management process:** Bring appropriate stakeholders and SMEs in at the beginning of the process, helping to decrease wasteful rework.
2. **Keeping the business analysts engaged in the development process:** Engage them more frequently and throughout the product lifecycle to provide valuable, timely feedback to the engineering teams, helping to steer teams more effectively before tangential work accumulates.
3. **Formalizing their standards to establish clear, well-controlled execution patterns and practices:** Initially created for the core product team, ultimately, this will be done for the entire organization.
4. **Developing standardized operational metrics:** These will help them objectively manage product lifecycle execution.
5. **Significantly ramping up their quality automation investment:** The goal is to massively decrease regression testing workload over time while increasing quality throughout the product lifecycle.

THE RESULTS:

Global surveys of the IT/development organization revealed that the development improvements increased satisfaction by 39 percent over the previous year regarding execution efficiency. Today, their software development process is now more efficient and executed with excellence.

In just this first engagement, they gained much clearer visibility into the state of their process and created a future-state model to represent their desired custom development lifecycle. Through our work, they could fully evolve the organization's process from idea to product release.

Additionally, through various evaluations of specific technical aspects of the product architecture, we designed a roadmap to achieve their future-state milestones and, through these improvements, help the organization expand their product to the United Kingdom.

Performance Enabled.

About SDLC Partners

Our commitment to our clients is to deliver customized digital solutions that transform the business through our uniquely enabled talent, processes, and leadership.

As a high-growth firm, we have garnered awards and attention from the Pittsburgh Technology Council, the Inc. 5000, the Pittsburgh 100, and E&Y's Entrepreneur of the Year.

Contact Us:

412.251.0848 or solutiondesk@sdlcpartners.com

Authors:

Ryan King, Vice President, Services