DEVOPS VS AGILE: A COMPLETE INTRODUCTION

Agile and DevOps are buzzy theories that many organizations are eager to employ, but there is often some confusion or overlap between the two. Only one business practice originated in software development, though both are being applied to organizational functions beyond software development. They're not the same, but they're not enemies.

Using Agile and DevOps in tandem is often the best approach for affecting change within a team, department, or an entire organization. Understanding both theories means being flexible to how they are constantly changing, and realizing that there isn't one answer to solve all organization needs.

A brief history of software development

Let's step back a few decades. Starting in the 1950s and 1960s, software development was a relatively new, but rapidly growing, field. Initially, a defined approach for how to develop software didn't exist, but one organically took the lead: the Waterfall approach. The Waterfall approach holds that developers should first define customer need, and then move through development until they release a single, finished software product that meets the stated need.

It sounded good, but it didn't always work out. Over time, developers realized that often the customer's need changed as the development team built software for an outdated purpose. By the time a "finished product" was released to the client, the client needed something different altogether. This mean that both time and money are spent backtracking, reorganizing goals, and

throwing out pieces of the development that aren't needed any longer.

Developers also realized that they were siloed away from other departments that dealt with the customer – there was no "team" approach of working with marketing, designers, or others. This meant that developers were frequently not aware of changes or feedback from the client. This problem, and the Waterfall approach, generally leaves us in the 1990s when developers widely experimented with other approaches.

An agile approach

As software developers realized the immediate drawbacks of heavyweight processes like the Waterfall approach, they looked for approaches that made tweaking software a lot easier and more agile. They also wanted to provide more opportunities for end user feedback, ensuring they were on the right path forward. During the 1990s, many lightweight theories and principles for software development emerged, including the popular Scrum and Kanban methods.

In 2001, the *Manifesto for Agile Software Development* codified several of these theories and placed them all under the umbrella of Agile software development. Agile software development, or simply 'Agile', values the following:

- **People**. Teammates, customers, and interactions between these people—instead of processes and tools
- **Immediacy**. Working software—instead of comprehensive documentation
- **Flexibility**. Responding to, and even embracing, change—instead of following a predetermined plan

Agile also destroys the idea of a "finished product", which was the goal of the Waterfall approach. Instead, Agile believes that software development is iterative and incremental. With each new release of software, the customer can perform new functions or improve upon existing functions.

Agile methodologies encourage developers to break down software development into small pieces known as "user stories". This highlights the value Agile places on the customer, which helps the developers by providing faster feedback loops and ensuring product alignment with market need. Agile further advocates for adaptive planning, evolving development, early and continuous delivery, and continuous improvement so that developers can rapidly and flexibly respond to change in client needs, software, or other external factors.

The separate world of DevOps

While Agile was a response to waterfall methodologies, DevOps was not a response to Agile. The two theories actually aren't the same thing—until companies started seeing similarities and increased success and productivity when using them in tandem.

As IT became essential to businesses in the 21st century, two imperative areas emerged: IT Operations (ITOps) and Development Operations (DevOps). Responsibilities of ITOps includes ensuring security, compliance, and reliability, whereas DevOps is responsible for developing and deploying new products to the end-user. While ITOps ensures safety and security for all business needs using the network, DevOps walks a line between flexibility and the rigorous testing and communication that comes with deploying new software.

The merging of Agile and DevOps

In the last decade, companies have begun spinning off a specific DevOps team from their original IT team, or adding Agile approaches within their software development teams. Through these organization changes, several similarities between the two theories emerged.

Agile teams rely on automated build, test automation, Continuous Integration (CI) and Continuous Delivery (CD). DevOps teams often use all those tools and more, including the addition of configuration management, metrics and monitoring schemes, virtualization, and cloud computing.

For software developers who were frustrated by the shortcomings of a Waterfall approach, Agile felt like a whole new world. But Agile wasn't perfect either. Common drawbacks to Agile planning include missed deadlines, completed software components that are incompatible with each other due to separated scrums, or teams, and new features breaking old functions – which is a direct result of missed cooperation with DevOps and ITOps. One thing linked all these problems with Agile development: lack of communication.

This is where DevOps begins to fill the gap. DevOps is a theory rooted in communication, both within itself, as the developers and operators have to coordinate, but also across other departments. DevOps frequently communicates with ITOps to ensure secure and stable environments for testing, and their crossover to other teams like marketing and customer service makes sense as they deploy new software.

Proponents of using both theories in appropriate business needs believe that DevOps can be seen as an extension of Agile. Agile relies on cross-functional teams that typically include a designer, a tester, and a developer. DevOps takes this one step further by adding an operations person who can ease the transition from software to deployment. Because of DevOps' inherent communication with other teams, DevOps can help automate processes and improve transparency for all teams.

Agile vs DevOps: Contrasting points

While we are proponents of using Agile and DevOps theories together, it is important to understand where they clearly differ:

- **Speed.** Agile is all about rapid and frequent deployment, but this is rarely the goal, or even part of it, for DevOps.
- **Creating vs deploying.** Developing software is inherent to Agile, but DevOps is concerned with the appropriate deployment of said software. For the record, DevOps can deploy software that was developed in any number of approaches, including Agile and non-Agile theories, like the Waterfall approach, which is still appropriate for certain projects.
- **Specialization.** Agile is an equal opportunity team: every member of the scrum can do every job within the team, which prevents slowdowns and bottlenecks. DevOps, on the other hand, assumes separate teams for development and operations, and people stay within their teams, but they all communicate frequently.
- **Communication.** Daily, informal meetings are at the heart of Agile approaches, so each team member can share progress, daily goals, and indicate help when needed. These scrums are not meant to go over documentation or milestones and metrics. DevOps meeting are not daily.
- **Documentation.** Agile teams don't codify their meeting minutes or other communications, often preferring lo-fi methods of simple pen and paper. DevOps takes documentation

seriously, requiring design documents and specs in order to fully understand a software release.

- **Team size.** Staying small is the core of Agile: the smaller the team, the fewer people on it, the faster they can move, even if they are contributing to a larger effort. DevOps will have many teams that work together and each team can realistically practice different theories.
- **Scheduling.** Agile teams work in short, predetermined amounts of time, known as sprints. Sprints rarely last longer than a month, and often can be as short as a week. DevOps values maximum reliability, so they focus on a long-term schedule that minimizes business disruptions.
- **Automation.** This is the heart of DevOps, as their overall goal is to minimize disruptions and maximize efficiency, especially when deploying software. Agile doesn't require automation.

These stark differences remind us that Agile and DevOps, at their roots, are not the same.

Culture of Agile and DevOps

While Agile does not necessarily lead to DevOps, both can have profound culture shifts within an organization.

An Agile approach encourages a change in how we think about development. Instead of thinking of development as cumbersome, Agile thinking promotes small, manageable changes quickly that, over time, lead to large change. Companies of all sizes have experimented with how working in an Agile way can boost many departments. Today some enterprises consider themselves fully Agile.

DevOps can also bring its own cultural shifts within an organization, including enhanced communication, and balancing stability with change and flexibility.

Choosing to use both theories is an active decision that many industry experts believe can lead to more rational decision making, thus improving the company culture.

Additional resources

DevOps vs Agile from **Edureka!**