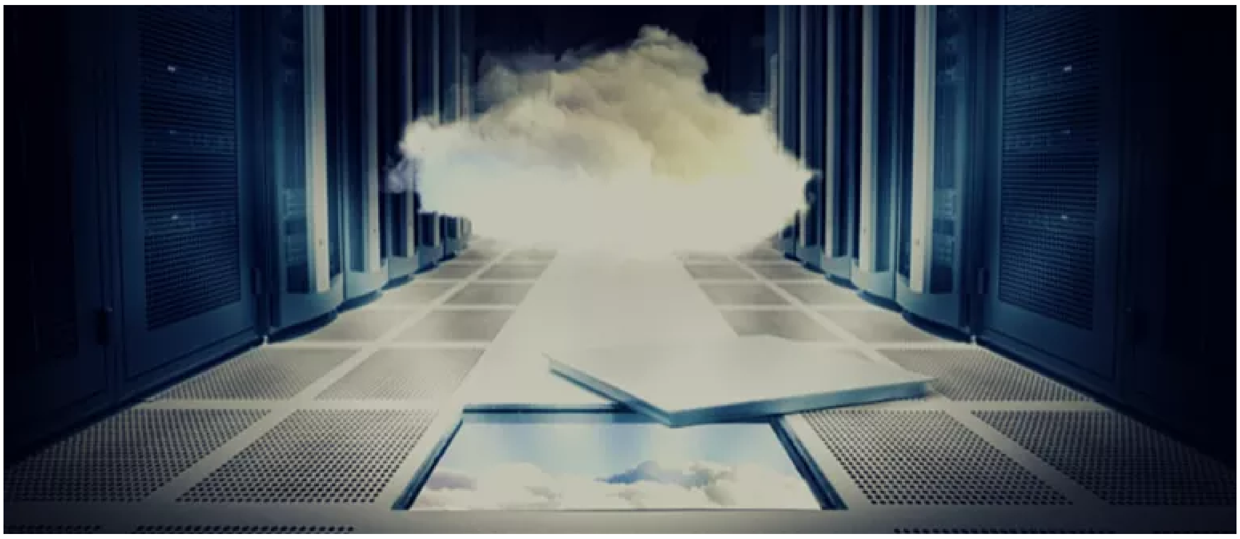
Comparing DevOps to traditional IT: Eight key differences

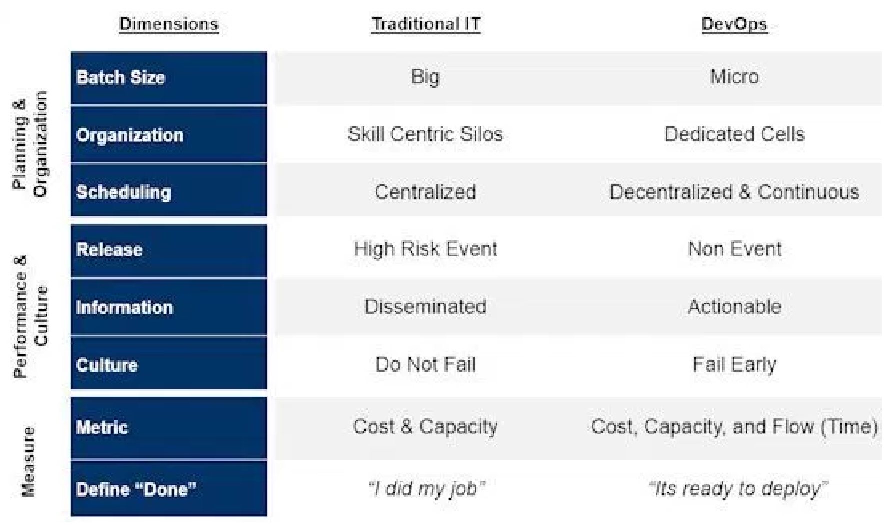
BY CONTRIBUTOR ON FEBRUARY 17, 2015 — 2 COMMENTS



You can make the argument that in the Software Delivery Life Cycle (SDLC) value is created in two places. First, when the software is created (development) and second, when the software is delivered to the customer and feedback received. The rest is transportation. The faster one can move code from the hands of the developer to the customer and receive feedback, the more value can be realized by both. Don't get me wrong, there is benefit to various "transportation" activities (testing, QA, staging etc.). But those are not value creating activities, they are risk mitigating processes.

A DevOps organization understands this value equation. They focus on the processes that add value, organize themselves around those actions and do the best to minimize the risk. Traditional IT gets it too – at least conceptually. However they treat all stages in the SDLC with equal importance or worse emphasize the wrong stage(s).

In general, when comparing DevOps organizations to traditional IT, DevOps will have policies and practices that differ from those of traditional IT along the following eight key dimensions,



	Dimensions	Traditional IT	DevOps
Planning & Organization	Batch Size	Big	Micro
	Organization	Skill Centric Silos	Dedicated Cells
	Scheduling	Centralized	Decentralized & Continuous
Performance & Culture	Release	High Risk Event	Non Event
	Information	Disseminated	Actionable
	Culture	Do Not Fail	Fail Early
Measure	Metric	Cost & Capacity	Cost, Capacity, and Flow (Time)
	Define "Done"	*"I did my job"*	*"Its ready to deploy"*

Planning & Organization

1. **Batch Sizes: Go from Big to Mirco**

Traditional IT has a bias for going big and for good reason. First, most development shops grew out of the waterfall method which by its inherent nature takes a lot of time. Second, since releases are costly and disruptive, operations allows developers only a few windows a year to release software. And third, its simply not sexy – doing big is how you get promoted. As a result, development organizations maximize productivity, by planning big projects, that involve a lot of code, bundled (hastily at times) into a release, and jammed into production.

A DevOps organization takes the opposite point of view and believe that small is beautiful. They understand that large batch sizes are inherently complex, risky (since there are so many moving parts), and hard to coordinate. Small batches sizes, on the other hand, are simple, easy to understand and test rigorously, and less risky. If things go wrong the impact is minimal and it is much easier and faster to fix. In other words, by going small organizations are able to perform more frequent releases and become more responsive to the customer.

2. **Organization: From Skill Centric Silos to Dedicated Cells**

Traditional IT is organized around skill centric silos. For the most part silos work, they band like skills together, drive greater utilization, and benefit from economies of scale. However, where these "cost optimized" silos break down is at the hand offs. In a typical IT environment, a new feature has to go through at least 3 – 4 silos before the customer gets it. It is not uncommon for an idea / code to spend 80% of its time waiting or ping-ponging back and forth between silos.

A DevOps organization also operates in a silos but at a different cross section. Here teams are arranged in "cells", consisting of dedicated cross functional teams, and focused on only one application. By creating this self-sufficient cell that consists of developers, testers, business analysts, and operators, an idea can move from one stage to another without hand offs, promotes cross-training / understanding (key to collaboration), focuses the team on the end goal, and encourages the "shift left" thinking. As for utilization and scale benefits, the rules for response states that for every ¼ reduction in cycle time, productivity will improve 2x, and operating costs by 20%.

3. **Scheduling: Centralize to Decentralize & Continuous**

Efficient scheduling is at the heart of a Traditional IT organization. Since resources are pooled, projects are usually clamoring for access to SMEs and / or infrastructure. To get around this, enterprises have invested in sophisticated scheduling planning systems. And while these systems are quite sensitive, they are inherently inaccurate, take up to much time to manage, and in some cases become the very bottleneck that they are trying to alleviate.

In a DevOps organization scheduling is pushed to the local cell level. The combination of smaller batch sizes, dedicated teams, and automated processes makes scheduling more simpler to operate. How? First, forecasting is limited to the very near future (2 – 3 weeks) where the teams have better insight. Second, there is no fighting for people's time since it is a dedicated team. Third, there is no waiting for infrastructure since it has already been defined and automatically provisioned. And fourth, it eliminates time consuming loop backs to management for escalations and decisions.

Performance & Culture

4. **Release: Turn a High Risk Event to a Non Event**

In a Traditional IT organization, releasing software into production is a high risk proposition. It is fraught with issues, escalations, and constant fire-fighting. The process is tightly managed, governed from the highest levels, and requires participation from all parts of the organization. It is not uncommon for IT to set up war rooms staffed 24×7 for weeks before and after the release.

DevOps organizations, on the other hand, make the release of software into a non-event much as possible. They reduce risk by daily integrating code into the trunk, automating testing, ensuring all environments are in sync, reducing batch sizes (as mentioned above) etc. In other words, they only promote code from one stage to another after they are confident that it will work in production. Thus eliminating all the hoopla about release windows and they are able to move new functionality into production at a much faster clip.

5. **Information: Focus is on Dissemination vs. Actionable**

Both types of organizations generate and share a ton of data. The difference lies in how the teams uses data. In Traditional IT, information is generated by specialists (e.g. operations team), bundled together with other data into a massive report, goes through management approval, is then shared with other managers, who then send it out their specialist (testers, developers etc.). In most cases the report goes unread, simply because there is too much data, not timely enough, and / or not the right data. In other words, information is shared but poorly consumed, and rarely used to take any actions.

Within a DevOps organization, it's the team cell that gather and creates the data. Since the data is to be consumed locally they only collect the data (also automated) that the team deems is necessary. And since the data is processed within the team, it eliminates the time lag of creating lengthy reports, manager approvals, and queue time (sitting in some ones email box). As a result, the team is quickly able to read and react to the data – resulting in faster feedback time.

6. **Culture: Do Not Fail vs. Fail Early**

Traditional IT is fundamentally a risk averse organization. A CIO's first priority is to do no harm to the business. It is the reason why IT invests in so much red tape, processes, approvals etc. All focused on preventing failure. And yet despite all these investments, IT has a terrible track record – 30% of new projects are delivered late, 50% of all new enhancements are rolled back due to quality issues, and 40% of the delay is caused by infrastructure issues.

A DevOps organization is risk averse too but they also understand that failure is inevitable. So instead of trying to eliminate failure they prefer to choose when and how they fail. They prefer to fail small, fail early, and recover fast. And they have built their structure and process around it. Again the building blocks we have referred to in this article – from test driven development, daily integration, done mean deployable, small batch sizes, cell structure, automation etc. all reinforce this mindset.

Measure

7. **Metric: Cost & Capacity to Cost, Capacity, & Flow**

Silos work for Traditional IT because they are directly tied to its measurement model – cost and capacity. IT is measured on how much can it get done (capacity) for the least amount of money (cost). It is no wonder that cutting cost, while trying to keep capacity constant, has been all the rage in IT over the last two decades. It's one of the reasons why outsourcing became so popular.

And while this model works great for stable legacy applications, the newer "system of engagement" applications require an additional metric – the element of time. DevOps organization understand this paradigm shift and have added "flow" as an additional metric. Flow forces an organization to take a look at its end to end cycle time, identify areas of waste, calculate true productive time, quantify quality, and focus on activities that add the most value.

8. **Definition of Done: "I did my job" vs. "it's ready to deploy"**

Another source of measurement difference is in how the two types of organizations define done. In Traditional IT, done is defined as the specialist doing "just" their part and handing it off. In essence, their done is more focused on meeting the hand off deadline vs. making sure what is done is deployable. This varying definition leads to some bad habits such as sub-par work, quality loss, finger pointing, and an overall sub-optimized process.

To get way from the finger pointing mentality is another reason why a DevOps organization prefers to create a dedicated cross functional team. With every member of the various parts of IT represented in this "cell" structure and all of them being held accountable for one and only one thing, they all end up having just one definition of done – bring quality software to market.

If you would like to learn more about this topic, please join me at IBM InterConnect 2015. I will be hosting several workshops on DevOps.

About the Author/Mustafa Kapadia

Mustafa Kapadia is the Service Line Leader for IBM's DevOps practice, a business advisory practice focused on helping large enterprises transform their software & application delivery. He has over 17+ years of experience in the tech. space, both as a service provider and a management consultant.

Prior to joining IBM, Mustafa was a management consultant with Deloitte's Strategy & Operations practice. He lives in San Francisco Bay Area, is an avid blogger (when time permitting), a speaker, and adviser to start ups.

Connect with Mustafa LinkedIn/ Twitter

FILED UNDER: FEATURES, VARIABLE SPEED DEVOPS
TAGGED WITH: DEVOPS COMPARISON, IBM, INTERCONNECT