The DevOps Studio

Dr. Ian Mitchell, December 16th 2016

"I love the new technology. New things give you a reason to want to go to the studio. New challenges mean you have to keep up, you know?" - Dr. Dre

Going to Where the Work Is

In an efficient agile team, each member will be able to go to where the work is. They'll be cross-skilled and capable of doing more than one type of job. No one will be constrained to having a particular skill-set, or into performing a unique specialism.

When people can only do one thing, efficiency is lost. This is because, in effect, they have to wait for work to come to them. Someone else has to finish their job first, and time spent waiting is time lost. On the other hand, having "skill silos" like this may appear to be reasonable because it exposes a clear value stream. It is obvious what the roles and responsibilities are, and what sort of value is being added at each and every point. However, the inefficiencies of such an arrangement soon become apparent. There is no guarantee of work being done at the same rate by each person in the stream. The nature of the work is different since it involves different disciplines such as design and programming, deployment, and testing. This means that bottlenecks can build up in one skill silo and no one else will be able to pitch in and to help clear it precisely because they don't have the necessary skills. There will be an accumulation of partially completed work in progress, a depreciation in the value of that work, and reduced throughput to the customer or end-user.

Note that cross-skilling helps even when it is done in a small way. If team members are cross-trained to the point that they can help out their immediate neighbor in the stream, then the flow can be evened out and the risk of bottlenecks can be reduced considerably.

The DevOps Divide

Two such silos have long afflicted the IT industry: "development" and "operations." In other words, there have been those who work on development projects and those who are responsible for configuring, deploying, supporting, and maintaining systems once projects have finished. They are very large silos consisting of multiple teams grouped together.

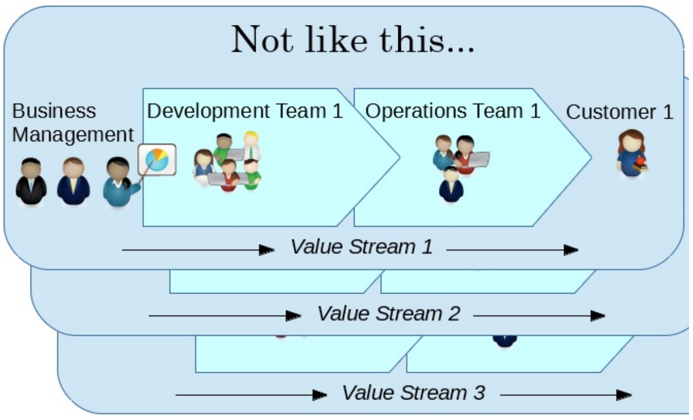
Work is handed off from development teams to operations so it can be put into production and maintained. Problems may be found at that point, and the system may have to be patched up before it even reaches the end user. With many lurches and false starts which wreck the plan and increase costs, the huge batch of work shifts like a creaking, melting, and shearing glacier from one group to the other.

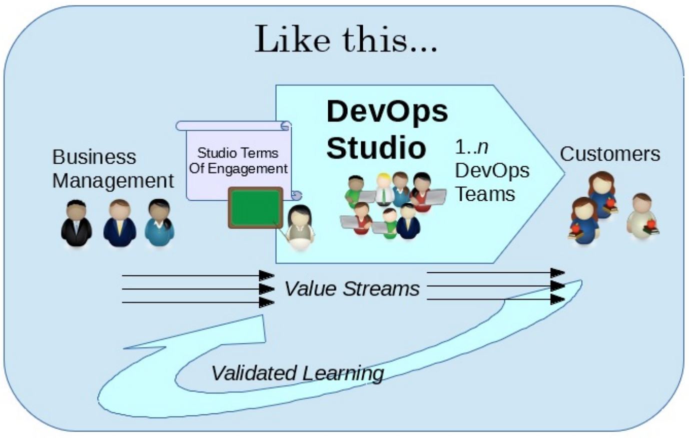
This arrangement has a lot to do with the way organizations traditionally fund such things. From the perspective of running a business, a divide between Development and Operations can seem to be rather arbitrary and wasteful. Both development and operational support work must be funded over a product lifecycle anyhow. The division can seem to be more an artifact of IT structuring than anything intrinsic to value delivery.

This implies that if budgets can be managed differently (and if development and operations teams can be trained to do each other's job) then perhaps the divide can and should be bridged. Seen in its most basic terms, this is the rationale of the "DevOps" approach. A DevOps team is able to go to where the work is, whether it be a development-related activity or to do with operational support and maintenance. Each team can support a particular value stream at any point in the product or service delivery lifecycle. Team members are in a position to work on programming or design, or configuration or deployment, without handover or delay. Furthermore, they can make good use of automation and tooling to expedite the entire value-delivery process, which they now own and can optimize further as they inspect and adapt it.

A New Model of Engagement

Not only that, but DevOps brings the potential for an improved model of engagement between Business and IT. Rather than setting up and tearing down IT teams on an ad-hoc basis, each serving a different business need as it arises, a DevOps Studio can be established. A studio brings one or more teams with a DevOps capability under a single roof, often in a quite literal sense. Irrespective of whatever the business need might be, at least one team is likely to be found in the studio which can do the required work efficiently across the value stream. There may be teams which can work on multiple value streams at the same time, by limiting work-in-progress for each one and thereby assuring throughput with respect to a specified quality of service.





In essence, work is organized vertically in terms of the value streams that are supported, rather than horizontally in terms of IT skill-silos and the usual funding conventions. With a DevOps studio in place, business managers can expect to deal with a professionally run one-stop shop for IT value creation and delivery. In turn, each DevOps team benefits from being part of the studio. Collaboration between teams is enhanced, with improved abilities to self-organize and to share best practices and lessons learned. They can also benefit from representation at this scale. For example, the studio may set defined and standardized terms of engagement which stipulate how the relationship of any team with business stakeholders will work. This can take the form of an actual contract, or perhaps in a more collaborative spirit, it can set out some basic rules or agile values that ought to be observed.

Example: The Rules of the DevOps Studio

1. Transparency, transformation, and innovation are to be embraced and not feared.
2. Testing, automation, and validation are the principal means to eliminate waste.
3. Tools, roles, and practices will scale to meet the challenge of ongoing delivery.
4. There will be a briskness to the work done here, but it will never be rushed.

A Note on Metrics

In a DevOps studio, empiricism is highly valued along with the metrics that support objective assessment. The most important metrics are actionable ones which support validated learning about an evolving product or service fit. Each increment delivered by a studio should thus be framed in terms of a testable hypothesis. Examples include customer engagement rates based on split-tests, increases in certain transactions, or the number of help-desk calls. Other metrics, such as throughput, latency, and cycle time can often be easier to measure, but they can also mean little without empirical evidence of stakeholder value being delivered in the first place.

Conclusion

The DevOps Studio is, in effect, established as a separate organization within the larger one. At least one Scrum Master, or person in an equivalent servant-leadership role, will be needed to set up the studio with its automation, continuous build and deployment infrastructure. They will also need to coach the best practices for studio operation. Since the team members of a DevOps Studio are expected to run it as an agile and self-adapting unit, the more the studio is used over time the better it will become. Nevertheless, it can be implemented to run in parallel with existing IT structures if organizational sponsorship and support for wholesale change are absent. In other words, it can be used to support a bimodal IT strategy where business stakeholders are allowed to make their choice. Stakeholders can continue to use the old IT capability if they wish and accept the risks of whatever track record it may have or they can use the studio and benefit from a more agile way of working.