DevOps: An Idea of Change for the Future – Part 4

By Loren Lisk on March 18, 2015

So far, we've discussed the Way of Flow and the Way of Feedback as being key to creating a system of information that helps the business create a process and improve communication. Our fourth and final installment discusses the Third Way, that of continuous learning.

The Third Way is the Way of Continual Experimentation and Learning

When we have achieved the first Two Ways, we can feel comfortable knowing that we can push the boundaries. We can experiment and fail fast, to achieve greatness. We have a constant feedback loop for each small experiment that allows us to validate our theories quickly. When we increase our Experimentation and Learning we see the following:

- We fail often, sometimes intentionally, to learn how to respond properly and discover where our limits are
- We detect faults in the production system as early as possible in the delivery pipeline
- We practice for outages and learn innovative ways to deal with them
- We push ourselves into the unknown more frequently, becoming comfortable in the uncomfortable
- We innovate and iterate in a "controlled" manner, knowing when we should keep pushing and when we should stop
- Our code commits are more reliable and production ready
- We test our business hypotheses at the beginning of the product pipeline, and measure the business results
- We constantly put pressure into the system, striving to decrease cycle times and improve flow
- We value resiliency over stability, since both external environments and internal structures for accomplishing things are complex and ever shifting, failure is "always around the corner"
- It should be treated as just another expected event rather than as an exception
- We value minimizing Mean Time To Repair (MTTR) over maximizing Mean Time Between Failures (MTBF): the inevitability of failure makes trying to maximize MTBF a futile exercise. Instead the focus should be on maximizing one's ability to repair failures. The dynamic nature of the market means that even working applications quickly fail to match shifting requirements. Not only operations but also development becomes an exercise in minimizing MTTR.

The most important takeaway here is actually that DevOps is not a toolset or a product, but a culture change to iterative, fast-quality improvements, where you smooth the flow through the company and each iteration. The result is that development gets faster and more efficient, until you have the best possible solution to provide your business deliverables to the customer in the most efficient and cost effective way possible.

Although failure is inevitable, it is better to speed the recovery time than trying to accomplish 9 9's uptime.

Flow will only improve if you define the value flow first, then look for improvements once you understand the reason the flow moves the way it does currently.

Small but exponential changes are how the most value it achieved.