The Art of DevOps Explained

By Rich Cefola on *March 27, 2018*

What is all this talk about DevOps and why should we care? Good question, and to answer, we need a little history of software development and its roots in traditional engineering principles.

Software development takes most of its predecessor processes from traditional engineering principles, which include drawing blueprints by skilled draftsman, having those evaluated by manufacturing teams, building tooling and making prototypes, refining the drawings and then manufacturing the product – over and over and over again in mass quantities. Skilled resources in each area throwing things over a wall before it finally gets released. Software development was no different – create detailed designs, write some code, test it and then release it as a product. The kicker is, software isn't really mass produced – there is essentially one source code, that everyone shares, so it is more like a sculpture that we all get to enjoy – therein lies the shift in thinking.

Change in hard goods is a major undertaking: tooling, manufacturing lines, supply chain. Change in software is relatively easy. Write a few new lines of code, change an image or two and a new database field. For the most part, it is just refining the sculpture. Software was being developed with a mountain of unnecessary overhead. Including packaging and deploying it to a production instance. Tech leads would create a slew of environments to try to corral change for almost no reason other than create a mountain of overhead and wasted hardware. So, throw away some of the unnecessary process, pitch the dev 1, dev 2, dev 3, test, integration, training, staging 1, staging 2, staging X, production environments and gain efficiency.

That is what DevOps does. It moves away from IT silos of system admins, developers, database admins and integration engineers, and develops cross functional teams that all work collectively on the sculpture. Refining it, adding to it and organically growing it. It enables Agility to live and breathe in the software development lifecycle through automating the deployment and testing to what can be termed "Blue/Green" platforms (Martin Fowler has much to say about it). Deploy the code in Blue (staging) – if it works, turn it Green (production) and off it goes. Have a problem? Swap it back. Then automate the traditional monitoring of application health, performance tuning, elastic scaling and the operations part becomes manageable by the same team.

In the end the result is Dev (developers) and Operations (administrators) all in on one team, continuing to refine the art. At the end of the day, it is a shift from a little less the rigor of engineering and a little more beauty of artistry. In the next blog, we'll dig a little deeper into how new approaches in Cloud in particular Containerization continue to make that manageable.