PERCEPTION, REALITY, AND CREATING TOMORROW'S DEVOPS DBA

Belgian surrealist artist René Magritte said, "Perception always intercedes between reality and ourselves." Take a smartphone. It is far, far more than a phone. It's an all-in-one communications, entertainment, and retail centre. It satisfies modern demands for instant service, instant messaging, and instant gratification. People want it all and they want it now—immediate upgrades and improvements, new features, no bugs, and smooth transitions.

So what makes our customers any different?

Enter DevOps, which aims to combine software development and IT operations (ITOps), shortening development lifecycles by providing continuous integration and delivery (CI/CD) with high software quality. Complementing the Agile methodology, DevOps looks to deliver software better, faster, safer, cheaper, and happier than before.

Breaking down the barriers defining the silo mentality that plagued development for decades, DevOps is not only about adequate automation, sufficient scheduling, and tolerable tooling. It's also about a culture of communication, people, practice, and process. But is everything in the DevOps garden quite as rosy as the idea suggests? Here's where perception and reality can clash.

Imagine this conversation: "Update your app? No problem. Update your OS? No worries. Want a database change? Sure. Unload, drop, recreate, reload. Oh, sorry. you didn't say you ***didn't*** *want an outage…"*

Every database administrator (DBA) knows that you don't replace an entire database, or even a table for every schema change. IBM® Db2® has become far more capable of handling online schema changes with ALTERs and deferred share-level change reorgs. There's always that little outage though, isn't there? The dreaded switch phase. Now, there are tools available to help developers perform more software changes more frequently. We call this the "release velocity."

Database schema changes, however, do not traditionally take the same path. Database release velocity is generally far slower. It can take minutes to deploy a new copy of code. It can take hours or even days to unload, drop, recreate, and reload a multi-terabyte tablespace. This is the "deployment gap," or "velocity gap."

How do we solve it? Well, let's consider the wider world, and a bit of history. With traditional waterfall development, no real value is delivered until the very end of a project when code moves to production. With Agile, you can deliver improvements in smaller chunks more frequently, adding value incrementally over time—and start much earlier in the project lifecycle, instead of waiting until the very end of what can be a monolithic process. And when development processes are properly automated, which is encouraged in DevOps, the results can be less risky, even when teams put out releases more often.

You'll also hear the term "shiftleft" when discussing DevOps. The traditional development process moves through multiple stages from left to right, starting with requirements gathering and ending with deployment and ongoing support. The shiftleft mentality looks to perform processes that typically happen later, earlier in development.

Now, developers have a well-defined automated process or pipeline to deploy code from system to system. However, when there's a schema change involved, the process must stop. It becomes a manual process of contacting the DBA (perhaps through Remedy or a ticket system) and the DBA finding time to determine what the developer is doing, what has changed, will it impact my system, does it follow our standards, do I have time to deploy it, etc. Throughout all of this, the development process is on hold.

Of course, we still need the DBA, but their expertise and role need to be integrated into the DevOps process.

In DevOps, developers (including development DBAs) and operations (including production DBAs) need to work together to deliver changes at a pace that satisfies both operational availability **and** development targets. Don't invite DBAs to your scrums only when you need them to implement an emergency change to a development environment because they haven't got the authority to do it themselves and have stalled. They need to be involved from the very beginning of the project design, from Scrum Zero!

Databases are not the same as applications. Changes need to be made before code changes, otherwise your BINDs will fail. There is also a much higher risk to database changes because you are affecting user data. Do it wrong and there's a serious impact. Change code and you affect one application. Change a database and you can affect many other shared systems. In short, your DBA is the best person to give you a wider view of system dependency. For too long, developers have thrown database requests or poorly performing code over the wall to a DBA who does their best to make it worthy of production, and throws it, along with any database changes, over the wall to operations.

DevOps is about people, process, and tooling. We need a culture change so we can move at a

speed that our customers now expect from us. This culture change extends to the database level, involving the DBA from the outset—the *DevOps DBA*. And we need to trust and empower developers to make their own changes, as and when they need to. This requires new ways to treat database changes as code, and to integrate them into the automated deployment pipeline. And believe me, it's easier than it sounds—we just need to change our perception of "who does what and when" to create a new and more effective reality.