**Transcript of video “It’s all about DevOps!”**

I’m going to take a minute to describe what it feels like to be a high performing business when it comes to concepts like continuous delivery. For me, prime effects of continuous deliveries are around cycle time. How quickly can a business go from having had that idea to the idea adding value to the business? Businesses such as Etsy, the large US retailer, are deploying to their production environment 50 times a day. Amazon, according to their record, is deploying to their production environment every 11.6 seconds. These businesses are applying this change because it lets them experiment continuously on the product they’re offering. They’re delivering their IT at the agility requirement of the business. And because of that high performance or high velocity, they’re able to run experiment after experiment after experiment. They’re able to tune their production systems not based on the hypothetical requirement s they’ve dreamed up during analysis, but based on real feedback for how the customers behave in live. They’re able to run experiments on just a 4 x 4 grid to maximize revenue. Are customers more likely to buy if the button is blue rather than green? And they’re able to do that by very very cheap experimentation without regret. Very focused on metrics and on providing that to the business metric which affect the way which the business operates.

By doing that we’re drawing in a huge number of concepts that come from Lean. A lot of this is very similar to the change in manufacturing industry went through as they moved from large batch manufacturing to just-in-time manufacturing. We’re working in much smaller batch sizes and to do that we’re looking to automate and industrialize the process of software delivery. Drawing from Lean, we’re looking at the value chain what are he steps that an idea has to go through before the idea becomes realized as something adding business value in life. And within that we’re looking to automate the processes through config management, one-step build and deploy, all test automation.

These concepts that are enabling us to really be efficient about the way in which we are delivering IT. If you draw the analogy from the way in which we have historically worked: 30 developers write a chunk of code and 3 months later, test report is slow. In contrast, John checks in his code and within 24hrs the automated test execution has run and has told him that interface 217 is now slow. John knows what he was changing, he probably knows why he was changing it, hopefully has the code still on his laptop. So we’re really cutting into that overhead of triage for trying to identify defects in a large body of code.

We’ve done something else and that something else is more important. We’ve made it really clear that it’s the developer’s responsibility, in this case John, to produce code at a quality that meets all of the prescribed quality gates. By baking those in and providing very rapid feedback on any issues found, we’re making it very clear to the developer that it’s their job to produce code that meets those quality gates, not the test team’s job to find issues afterwards. We’re baking quality into the way in which we’re delivering. We’re not relying on mass inspection at the end. And I think that’s a really fundamental shift in the way which we think about testing software development.

Another feature of the businesses who are operating with high capability in continuous delivery is that changes are introduced in real-time and are really easy to back out. In practice I don’t think a huge number of our clients except in very specialist cases want to throw change after change in live but the ones who are doing it, the case studies of people working with continuous delivery actually report higher availability. Although the changes are far more frequent, each change is much smaller and typically each change is much quicker to revert. They are typically hidden behind a feature toggle so they are really separating that distinction between “we’ve put the change into the code base” and “we’re requiring that change to be live and visible to users”.

Those businesses are testing those changes in environments that are really representative of live. If I use Etsy as an example, every developer can build a representative production environment on their laptop using the same chef recipes used in the production environment. And that really means that when they’re doing testing; right back to unit testing, that testing is meaningful. Production can be expected to behave the same way. And this is a real theme as we start looking at things like DevOps and continuous delivery. That idea that we’re going to treat all of our environments to the same processes to the same tooling as production; meaning that when we’re applying that change, its’ representative - the testing is representative. And with that comes a huge focus on all-test automation. 50 deploys a day doesn’t come without having built in a whole series of prerequisites first. Those prerequisites require rock solid version control. They require automated build and deployment. They require that deployment to not involve an outage. You can’t take 50 outages a day. But they also require a significant investment in test- automation. The ability to be able to tell a developer within minutes or if-must-be hours after they’ve introduced a defect, drives a real focus on automated testing rather than manual testing and a real focus on baking in everything you require to get representative go-live into that automated state.

My final theme in continuous delivery is risk of change. When these businesses are introducing changes into live, they’re not big-bang rollouts to their entire customer base. Instead, they’re directly tested in very small samples via A/B testing. I don’t know if you’ve noticed but recently Amazon launched a new design for their homepage. That had been tested in live for a long while before the release. Not tested in live as in whether it met the functional requirements or whether some technical metric was being affected, but tested in live against the metrics that matter to them as a business. I would imagine the basket size, customer dropout rate, number of page hits per customer, were the sorts of things that they would’ve been measuring. And initially, on a miniscule portion of their customer base, over time, that would’ve grown until they’re comfortable to rollout to everyone. There’s a feature of these businesses measuring those business metrics, those business KPIs as part of their operations management solution. Certainly focusing on things like servers, I’m sure, but really the metrics that they’ll be watching are A/B testing or red/blue testing of the metrics which are driving their business performance and if the change is not meeting those or they see another unexplained drop-off rates they’ll be responding as a business to those flows. So we’re really talking about *business* working with *development*, working with *operations* either as a separate team working really close together or potentially in some cases a slightly more merged team.