
Welcome everyone to this session DevOps Test Alone. My name is bjorn boisschot and I work for CTG a consultancy company in Belgium.

You probably know Belgiums for its world famous Waffles, chocolat and beer. But did you know that we are also the country that invented DevOps?

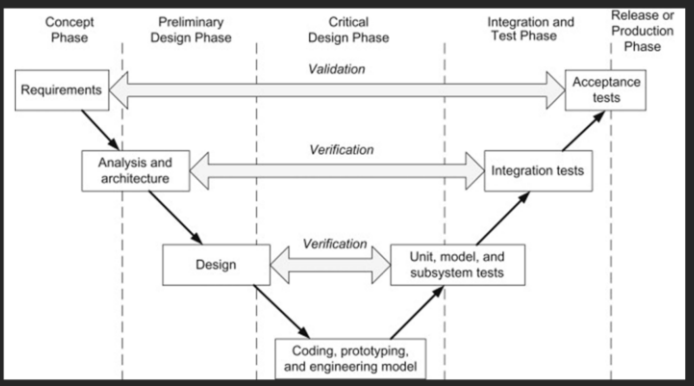
No, so lets start this presentation with some historical facts on testing before we


This was the year that Sony released the first ever WALKMAN. This was a milestone in the history of music as people were now able to listen to music when on the move!

It was also the year that the first book “THE ART OF SOFTWARE TESTING” dedicated on software testing was released,

This book has set the foundations for what ‘modern” software testing is all about today. He introduced concepts such as black box testing, the economics of software testing and many others.

# 1986




1986 was the year that Nintendo released their Nintendo Entertainment System on the world. This console has become one of the most loved consoles in gaming history. I remember spending quite some hours playing Mario Bros, Zelda and Metroid.

In 1986, the V-MODEL was introduced. This model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.

The V-Model is also the starting point for many companies to setup a more structured approach to software testing wehere the is now a dedicated point in time in the development lifecycle that specific testing effort is taking place.


2001: The human Genome Project released and was publicly accessible on the internet. This was a major milestone in discovering more on ourselves

In 2001 on a conference in Utah the Twelve Principles of Agile Software were published in the Agile Manifesto.

This was going to be a game changer for testing as quality was now not anymore for testing alone, it was owned by the complete team and was the responsibility of the complete team. Focus on test automation was also higher as the iterations to release software became smaller and the focus on quality became higher!


2009 was the year that the first bitcoin ever was mined and thus also the first transactions with bitcoins took place. Bitcoin has the potential to disrupt the way the entire world

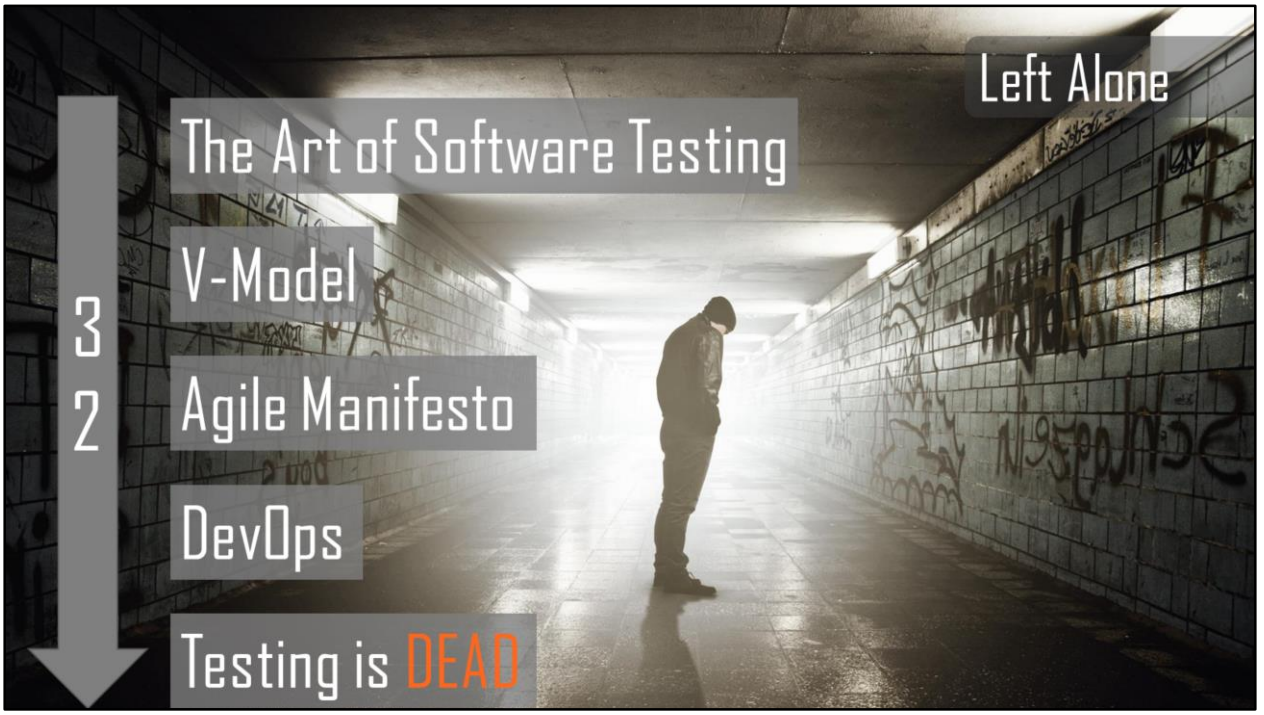
In 2009 the first DevOps Days conference was organized in the city of Gent (in Belgium).

The conference popularizes the software development method DevOps that stresses communication, collaboration and integration between software developers and information technology (IT) operations professionals.


2011 is the year that Apple's co-founder Steve Jobs died. Jobs was responsible of the disruption of multiple industries such as music, movies, computers, telephones, tablets, and many others.

In 2011, Alberto Savoia is the first one to state that "Testing is Dead". During his opening keynote at the Google Test Automation Conference 2011, Alberto Savoia states that (traditional) software testing is dead or on the decline.

08'




So in 32 years we went from the first dedicated book on our testing profession,

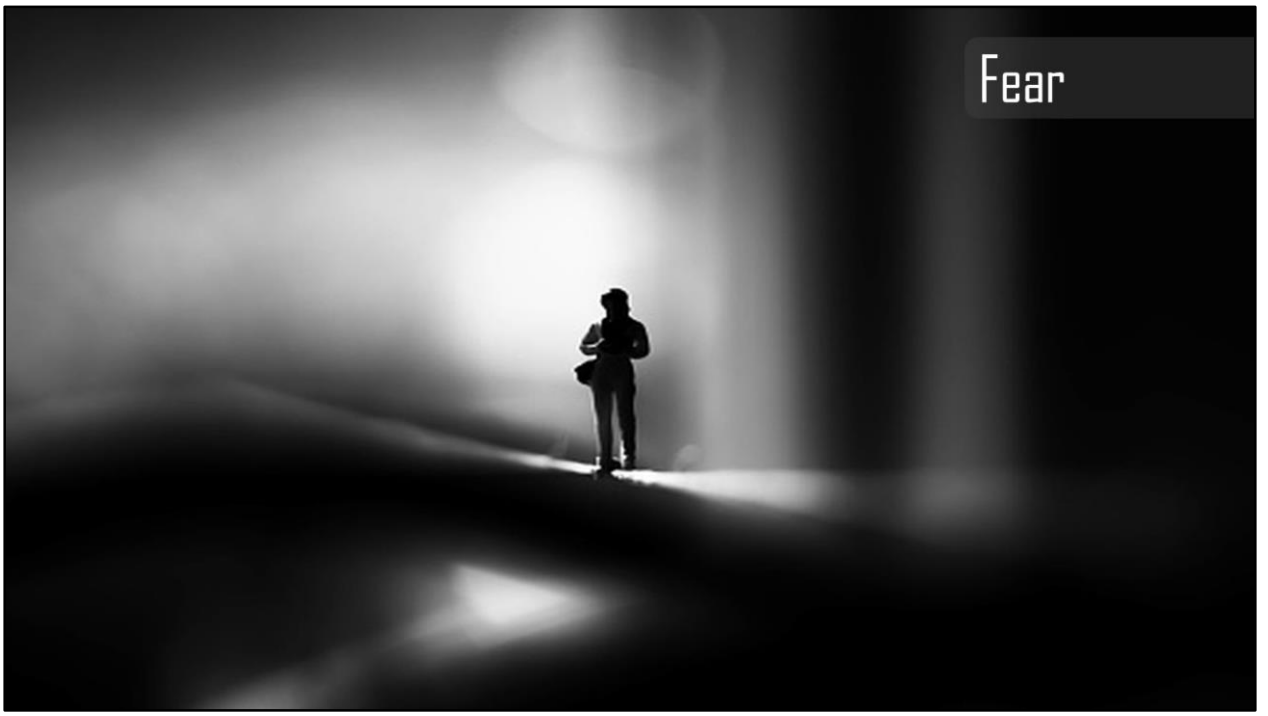
to a more structured process with the V-model

to more emphasis that testing is a team responsibility

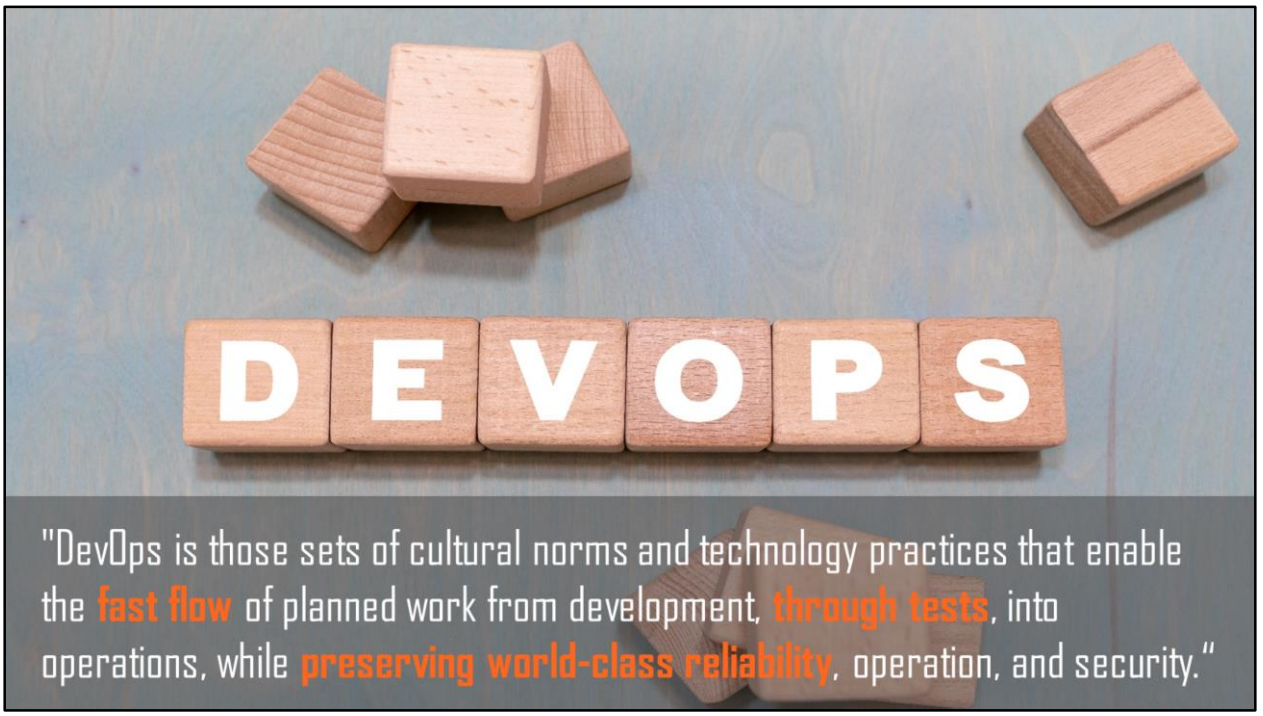
to the merger of the dev and ops teams

to the first announcement that our profession is dead!

This is quite a heavy journey we went through!



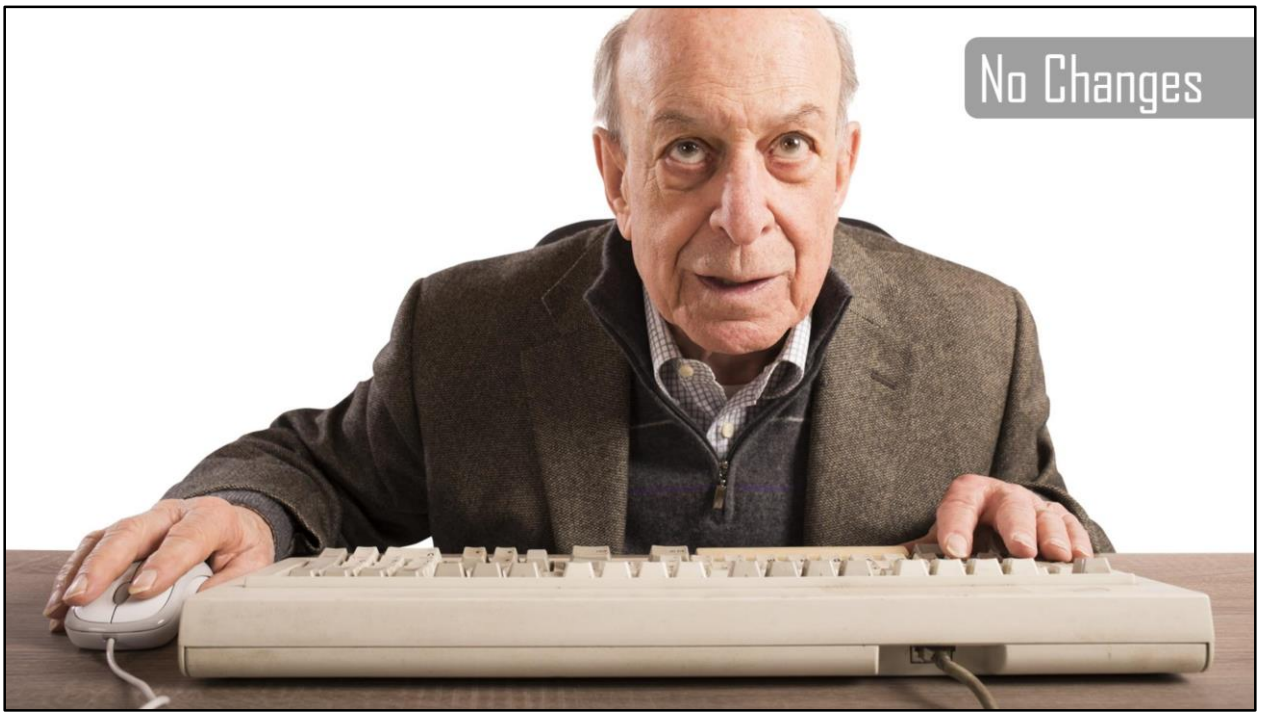

This was also the point when a lot of test professionals became worried for their jobs. We started hearing rumors from big tech companies in the devops world about the fact that they didn't need any testers anymore. More and more sessions on devops conferences talked about the fact of replacing testers by developers or bots (with our without AI).




So lets look a bit closer at one of the definitions of devops from Gene Kim (one of my beloved authors):

So DevOps is all about:

- Continuous Improvement
- Faster Releases
- Faster feedback
- Improved Quality
- Improving !!!VALUE!!!!




And what about testing…? What did we do the last 20 years in order to evolve our profession?

We have invested quite some more money and time in test automation (usually on the GUI).

We have matured some of our own testing processes.

But if you ask me, testing is playing catch-up and trying to understand how to adjust and adapt to this new reality

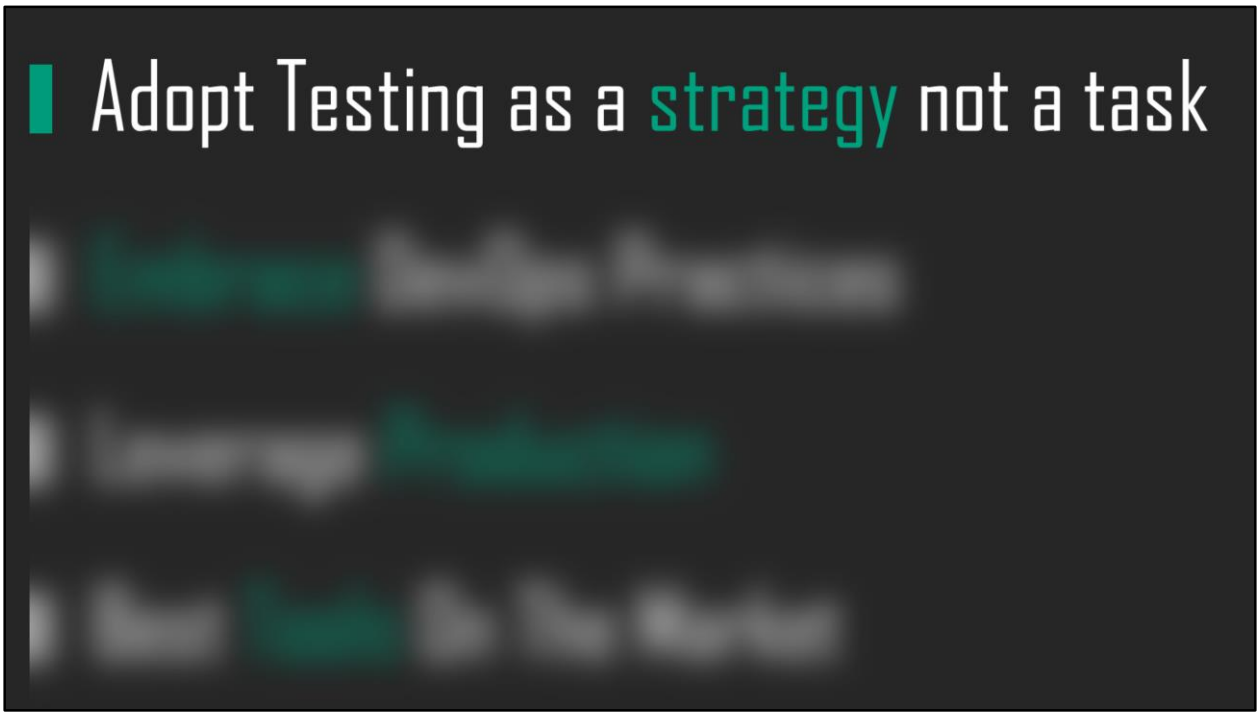
It stroke me when looking back at the talk “testing is dead”, in this talk James Whitaker stated that if we would take tester from the 70’s and asked him to test, that this wouldn’t be a problem for him! Our profession didn’t change during the last few decades!


Based on these facts we shouldn’t be surprised that people are questioning the traditional ways of testing.

Instead of remaining a re-active activity we need to ensure that we become a pro-active strategy!


For us it will become crucial to understand how we as testing can deliver value. Not only deliver value to the customer but really add value in the complete software delivery chain.

In some cases testing will even become a more supportive function to the others in the team.




We need to change the way people see testing. It is not a task performed by some individuals, nor a task performed by the whole team. Its should be looked at as a strategy!

10’


So, we need to take up the task to teach developers why testing is important and how they can easily contribute to it.

So when it comes to unit testing most developers only focus on the happy path. We as testers need to teach them to also look at negative cases, use boundary value analysis to determine the different possible inputs and outputs.

And we should be the ones that are reviewing their unit tests. I had one project in which the developers were creating unit tests and another developer was per-reviewing the unit test. The only remarks that were given were all on the technical implementation of the code, no remarks on the tests themselves! This is normal behavior as we both have different mindsets.

We can assist the developers on how to create performance tests, or take over that task. Certainly in the world of DevOps with API-first, micro-services and containerization it will become crucial to execute performance tests as soon as possible. And it is not even that difficult, you can write a small wrapper around your unit tests so they become unit performance tests. Next you will need to feed the results in some trend graphs and you will immediately see if your code has slowed

down the system. Loadtesting your API's can be done quite soon in the SDLC and is also something that usually remains quite stable. So these are all quick wins.

Last but not least we can help with the setup of the CI pipeline. Defining which automated tests to run in the pipeline, setting-up the different stages. Checking how we can speed-up the execution of the automated tests so we can have faster feedback. All tasks at which we more technical testers are quite good at.


The coaching of the developers will generally be done by more technical testers, but equally important is the coaching of the business.

We need to teach business on how to define good acceptance criteria, ensure that they create them SMART. Show them if they create good acceptance criteria that the software delivered is of better quality and does what they intended it to do. Make sure they understand that this effort will make their lives better and all of our jobs easier.

We should perform some peer reviews of the user stories delivered. Generally as testers we have a great view on how the real end user is going to use the software, so we are also the ideal candidate to review the user stories.

We can also help to understand and define the different persona's that are using our software. As testers we are great in putting ourselves in the mindset of the people using our software.


One huge important factor that most managers forget is that in order to have good collaboration you need an environment of TRUST.

Before we can truly work together with dev, business and ops, we need some level of trust.

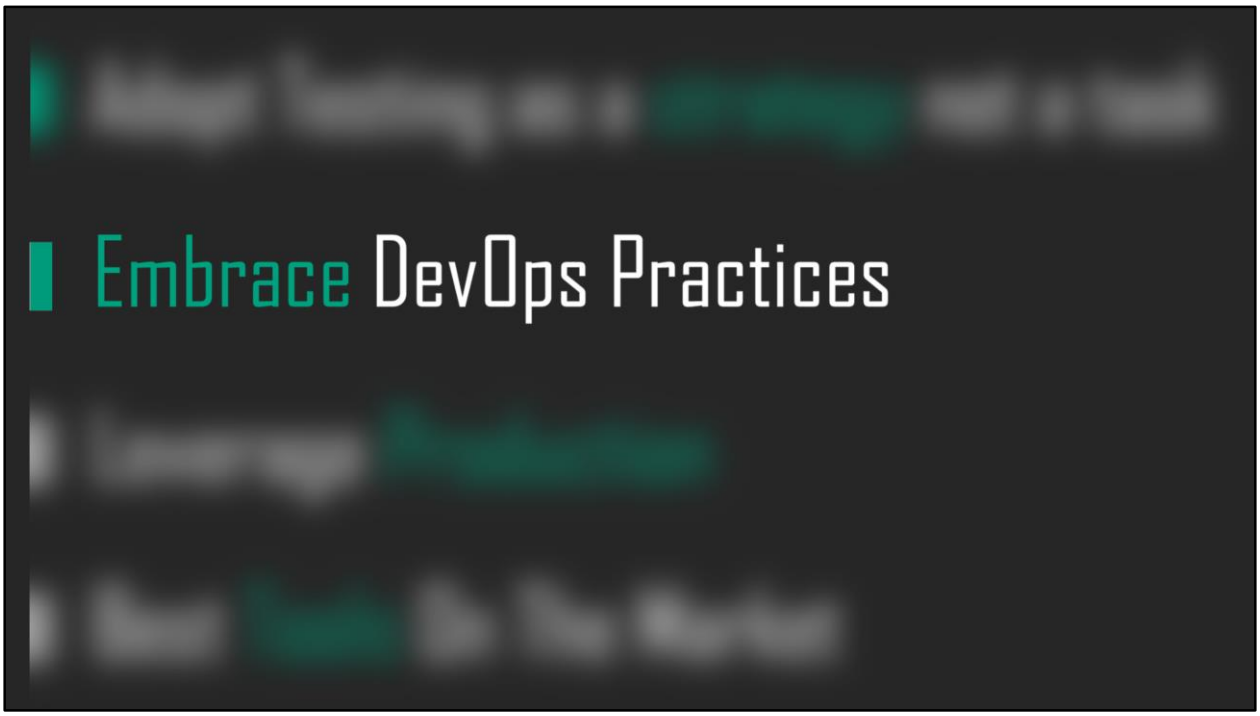
Trust needs to be cultivated, this can be done on the following ways:

1. COFFEE, COFFFE, COFFEE and cakes
    1. Go get a coffee and talk about non-work related things
    2. Bring food at any occasion
2. Have empathy with the developer
    1. When bringing bad news to the developers on the quality of the project, try to deliver this message with compassion! We are the ones that are going to state that their newly born baby is ugly and has some serious problems!
3. Be positive and celebrate success
    1. Congratulate them on their first unit tests
    2. Thank them when they find important bugs

3. Go out and get a drink each time there is a successful release

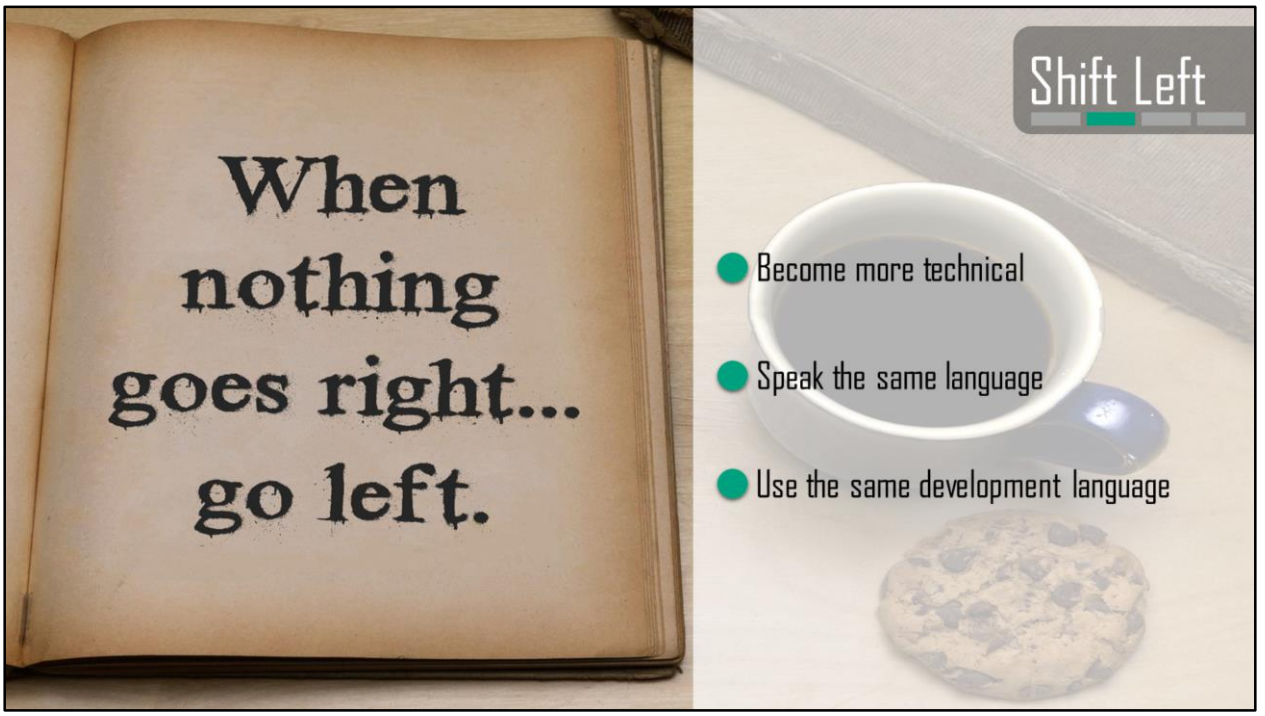
For most of the above actions we really need some good soft skills.

Soft skills are one of those skills that are harder to train than hard skills. So in my opinion this is something companies should embed deeper in their recruiting process and value much more than they are doing today.




As testers we don’t need to be afraid of DevOps, we should actually embrace some of the techniques and use these for our purposes!

20’




I know this is a hype word that is making a lot of testers nervous but I do believe that it is partially true!

We will need to become more technical, we need to understand the technical context of our application! We don't necessary need to become programmers and learn how to write code!

Remember one of the first slides about trust! This is one way of building trust with the developers.

As an example of that:

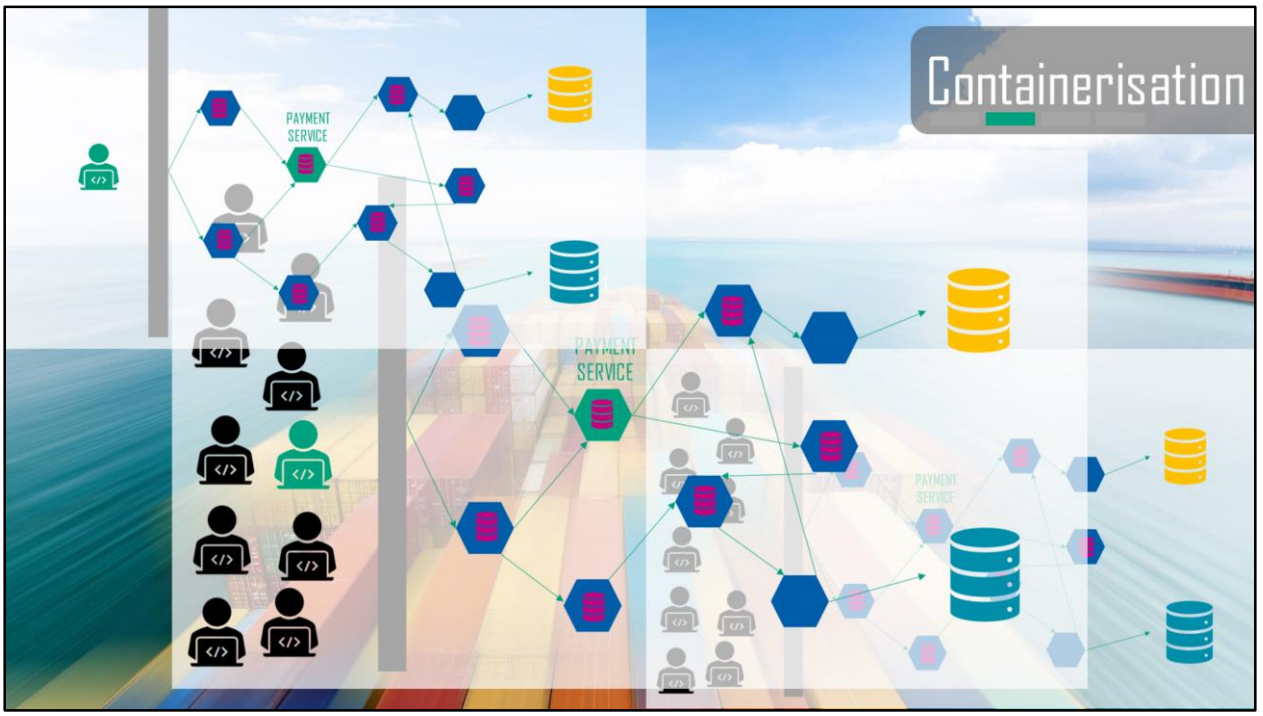
In one of the projects I was working on as a coach I noticed that the defects that were created by testers who were more technical and could check the application logs on the Unix servers, were able to figure out what the error stack traces was all about were fixed sooner that the ones reported by the more none-technical testers (despite of their severity).

When asking the dev team on this, they answered me that this was logical because these testers were talking with the dev team on the defects found. When the dev

team explained what was happening, the testers understood them. This actually resulted in a kind of human bonding between the dev team and these testers, resulting in favoring their defects!

So for this reason only I believe it is critical that we can talk the same language as the developers to ensure we become one team!

When creating test automation scripts or other non-functional tests, use the same language as the development team does. This will enhance collaboration; it will make it much easier to get assistance and the dev team will find it more convenient to also work on automating test cases.




Containerization is certainly one of the hot topics these days. As testers we can benefit a lot from a containerized setup of our application under test.

In the bank I was working for last year, they had created a new application that was composed of different micro-services all running on Docker containers. One of the important things for us to test was to see how the application was reacting when we stopped some specific micro services (figure out who would lose money in this case ☺). If we would do this in the test/acceptance environment, we would be hindering all of the other people working on this environment. Because of the containerization, we were able to spin up dedicated environments for each of the testers so they could destroy whatever micro service without hindering the others.

Another use case we had was managing the test data for our automated tests. Because of the containerization of our application, we were able to spin up containers, each with a dedicated database. So at the start of our automated test run, we provision our infra, startup the necessary containers and off we go. This way our automated tests could be rerun with every change of the source code.

This principle is not only applicable to micro-services architectures. Containerization

can be used for a wide range of purposes!

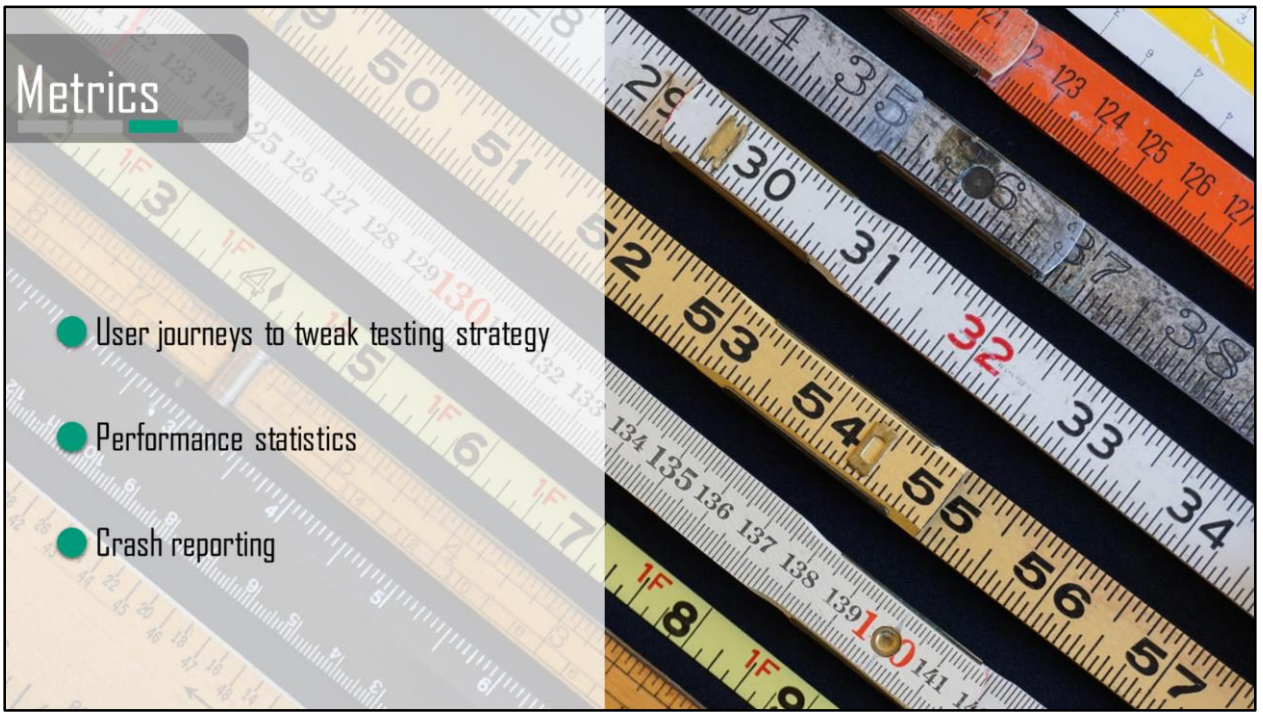

If we look at this bathroom from an end-to-end perspective, it looks great. It is functional, working and quite a nice design.


But if we look at what is behind this bathroom, the pluming….

We immediately notice that any problem or change will take ages. If one of the pipes get blocked, we will need a very expensive expert to fix this!


Sometimes it seems to me that we are afraid of our production environment. It is like a nuclear no-go zone that nobody can touch. When bringing the dev and ops team together, this should make it easier for us to get access to the rich world of production data.

30’




Earlier, testers used to rely on reported defects, surveyed shortcomings, reviews, and feedbacks to get this information from customers in production.

In this new age, thx to the merge of the Dev and Ops teams, we can access the data from production more easily!

Some examples from projects we did:

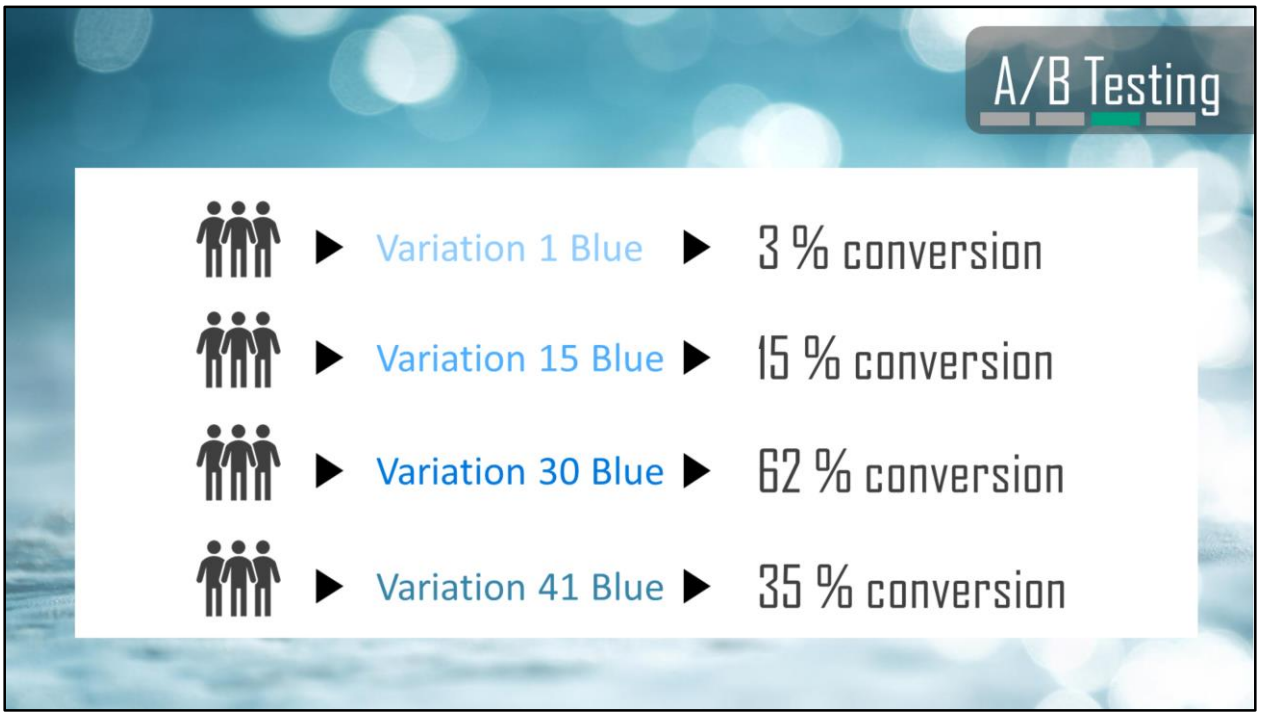
1. User Journeys: At one of our financial customers, we used the data collected by the Dynatrace APM solution to tweak and adjust the way we organized our system integration tests. The solution can help you trace a user's journey within your application by identifying which features users use first when they open the application, how long they spend using different features, and what they are doing before they close the application. What we saw was that the end-users actually followed a different navigation path than we did in our test cases. We also saw that some features we thought would be used a lot were actually not used that much. This allowed us to tweak our risk based testing approach and put less focus on these.
2. Performance statistics: Instead of creating performance scenarios based on what we think would be the usage scenario of our application. Look at the usage stats

in production. If you don't have an APM solution that does this, contact the Ops people and see how to retrieve this information from the different monitoring tools and logs they have. This was a real gamechanger for us as we could replicate the normal load of our systems quite accurately and could experiment to see what would happen if we got twice the amount of usage.

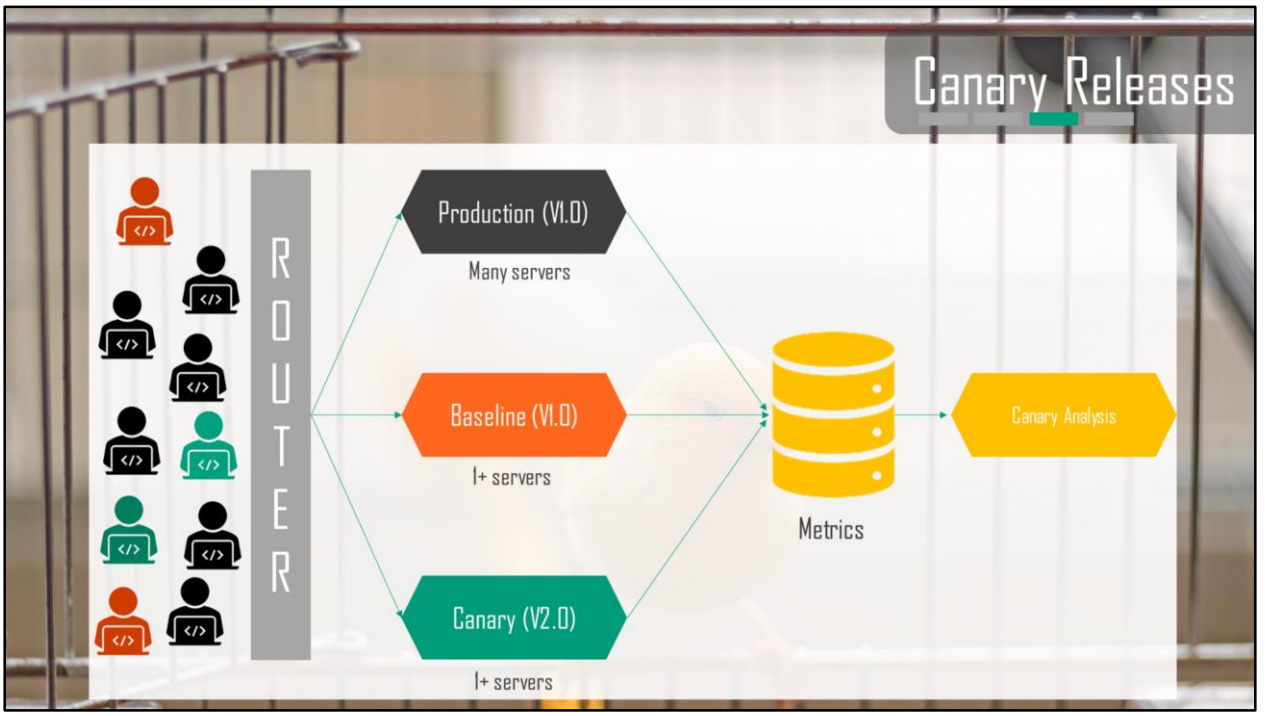
3. Crash reporting: One of our pharmaceutical customers has a solution in place that reports all the crashes/errors end users have when navigating their app. This report contains all the steps performed before the crash and detailed crash analytics. This is the best data you can imagine to reproduce the issue in the non-prod environment. Afterwards we actually used this on the TST environment during our exploratory testing sessions to figure out which actions we performed before encountering the bug.


To be very honest, this is not an area I have lots of expertise in but it is one I would love to be involved in.

This is the area where big devops players such as google, Netflix, Facebook and many others are experimenting with changes in production to see what the results are and to adopt to these.




One of the most know examples of this is the “40 shades of blue” A/B test that google did. 40 different shades of blue would be randomly shown to each 2.5% of visitors; Google would note which colour earned more clicks. And that was how the blue color you see in Google Mail and on the Google page was chosen.




Netflix is using automated canary releases to check if the latest release has the desired outcome and if it can be pushed to all the servers.

"A canary release is a technique to reduce the risk from deploying a new version of software into production. A new version of software, referred to as the canary, is deployed to a small subset of users alongside the stable running version. Traffic is split between these two versions such that a portion of incoming requests are diverted to the canary."

By tracking the usage of the feature, the log files and the feedback received, the company decides either to increase the rollout or to disable the feature, either improving it or discarding it altogether


A mandatory slide for any devops presentation, which are the best tools out there?


There are thousands of tools out there that can help you, which is the best is different for each company/project.

There is only one important thing to remember on tools: ENSURE YOU CAN USE THE TOOLS ON ALL THE PLATFORMS YOU HAVE DEV-TST-ACC-PRD, this is the only way to really benefit from a tool.

- Adopt Testing as a strategy not a task
- Embrace DevOps Practices
- Leverage Production
- Best Tools On The Market

In order to be able to adapt to these guidelines we will need to ensure that testing is involved in the early stages of the software development lifecycle!


All of the above should ultimately lead to built-in quality!

We should strive to ensure that quality is built in into the application and not an after-thought!

This is why I really love this quote by Edward Deming!

If there is one slide I want you to remember from my presentation it is this one.


TESTING IS
ALIVE
&
KICKING

# QUESTIONS?

TestCon
October 13 - 15

@bboisschot
Bjorn.Boisschot@ctg.com