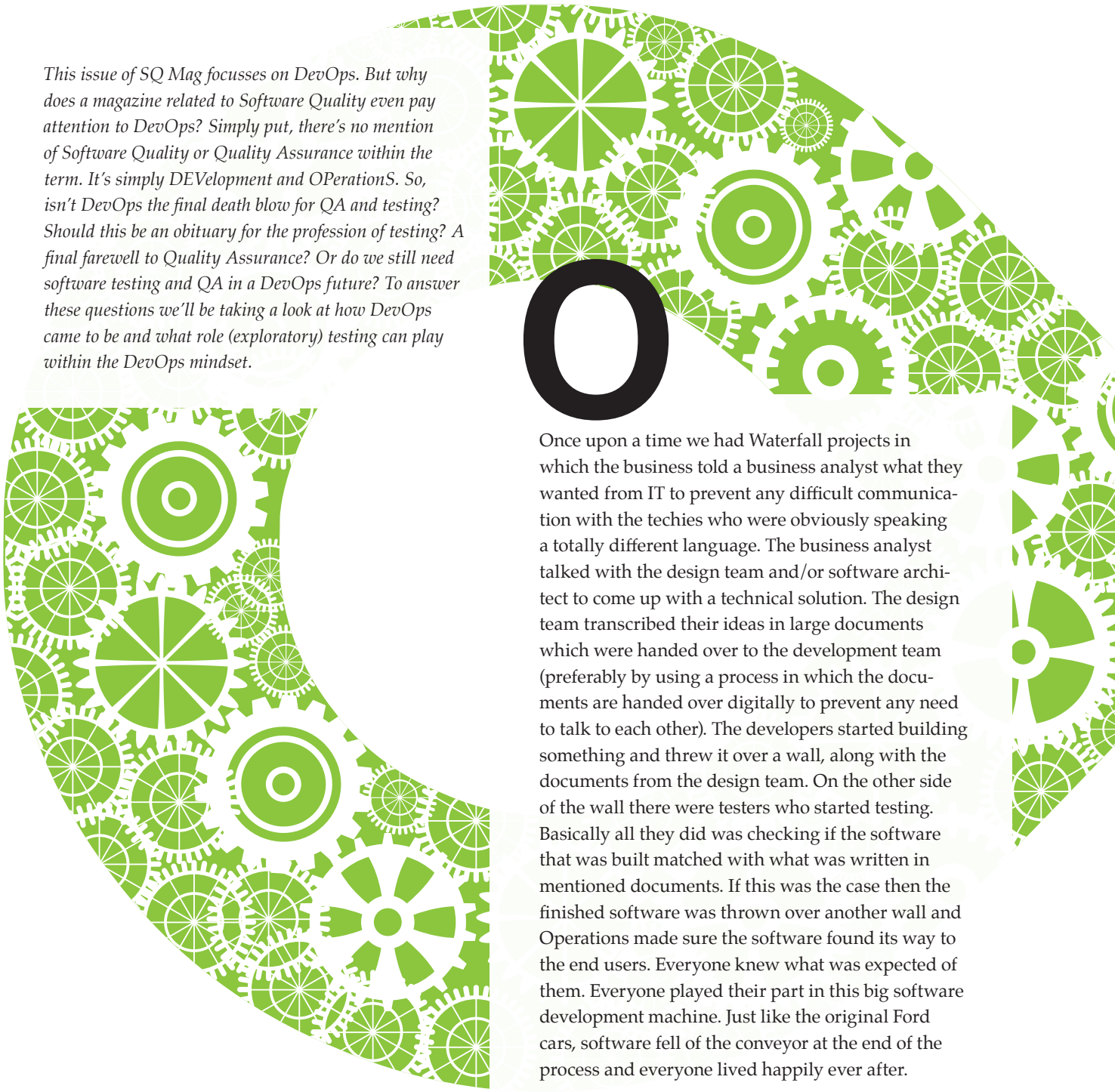
WHERE IS QA IN DevOps?

This issue of SQ Mag focusses on DevOps. But why does a magazine related to Software Quality even pay attention to DevOps? Simply put, there's no mention of Software Quality or Quality Assurance within the term. It's simply DEVelopment and OPerationS. So, isn't DevOps the final death blow for QA and testing? Should this be an obituary for the profession of testing? A final farewell to Quality Assurance? Or do we still need software testing and QA in a DevOps future? To answer these questions we'll be taking a look at how DevOps came to be and what role (exploratory) testing can play within the DevOps mindset.



Once upon a time we had Waterfall projects in which the business told a business analyst what they wanted from IT to prevent any difficult communication with the techies who were obviously speaking a totally different language. The business analyst talked with the design team and/or software architect to come up with a technical solution. The design team transcribed their ideas in large documents which were handed over to the development team (preferably by using a process in which the documents are handed over digitally to prevent any need to talk to each other). The developers started building something and threw it over a wall, along with the documents from the design team. On the other side of the wall there were testers who started testing. Basically all they did was checking if the software that was built matched with what was written in mentioned documents. If this was the case then the finished software was thrown over another wall and Operations made sure the software found its way to the end users. Everyone knew what was expected of them. Everyone played their part in this big software development machine. Just like the original Ford cars, software fell of the conveyor at the end of the process and everyone lived happily ever after.

↗ *3 Amigos*

You may have heard of the movie '3 Amigos', but not everyone (yet) knows of the term related to Agile. In short it's about a collaboration between three parties: Developer, Tester and Business (Product Owner). However, currently the term is often used for generally any collaboration/communication where Agile teams and/or other stakeholders are involved. Goal of the 3 (or more!) Amigos is to create a general understanding of what is needed, how this can be created and when it is considered done and of provable sufficient quality.

However, at around the turn of the century the fairytale finally shattered when end users began complaining (louder than before) that the software did not provide the functionality they so desperately needed. And in 'the old days' such complaints were hushed by telling the end user that technical possibilities weren't advanced enough to meet their requirements. And everyone accepted this. Currently technical debt is no longer an excuse to not deliver what the business needs. Computers are powerful enough and software has grown enough to meet most (if not all) requirements most everyday businesses throw at them. The software development conveyor began creaking and shrieking more and more. End users blamed ops, ops blamed testers, testers blamed developers who in turn pointed to the design documents that didn't tell them what the business really needed. Or at least, they hadn't been able to interpret them that way. Things had to change. And so they did. In 2001 a group of key figures in the industry locked themselves up in a ski resort in Utah and a few days later they emerged to the outside world again carrying the manifesto that changed the software development business forever: Agile was born[1].

Business actually talked with developers and testers (it eventually even earned itself a name: the 3 Amigos principle[2]). Walls were broken down. Software development became a collaboration process in which everyone was involved. Right up until the moment when the software was done and needed to be promoted to production. Then it was thrown over one last wall to Ops. Everyone wished them good luck with it and the next sprint was started in which new potentially shippable products were designed, built and tested. And what happened on the other side of this final wall was of no concern for the Agile team.

Strangely enough the Agile principles don't seem to apply for Operations. It's clear that this can cause huge issues when delivered shippable software is promoted to production the first time. The software may react differently than expected once implemented, there may be more work involved in deploying the software and/or maintenance might turn out to be very time-consuming and/or error-prone.

[1] http://agilemanifesto.org/ (2016-12-27)

[2] http://blog.gdinwiddie.com/2009/06/17/if-you-dont-automate-acceptance-tests/ (2017-01-24)

WHERE IS QA IN DevOps?

↗
Kaspar van Dam,
Consultant at Improve Quality Services, The Netherlands. With over 10 years of experience in IT, Kaspar advises colleagues and clients on matters concerning testing and/or collaboration and communication within (agile) teams. He has published a number of articles on test automation, agile ways of work and Continuous Communication and is a speaker on these matters at events.

You can follow Kaspar Linkedin: https://nl.linkedin.com/in/kvandam and contact him at Kaspar.van.dam@ImproveQS.nl

Enter: DevOps.

With DevOps this final wall was broken down[3]. Operations became part of the software development process and the (rest of the) Agile team became part of software delivery and maintenance. A simple principle applied: eat your own dog food[4].
Knowledge of the finalized product and how it behaves in production became an essential part of the iterative development process. DevOps teams have two choices: keep building software that won't (entirely) do its job in production and get to a point where you spend all available time on keeping the software afloat. Or build better software that requires less maintenance and free up time to keep creating even better and more innovative software. Needless to say, most teams went for this last option. Basically we went from the Ford conveyor belt to the Formula one pit street where every team member is intimately involved to win the race.

Now, back to the main question of this article. Where is QA in DevOps? The term itself already implies DevOps is an effort mainly by Development and Operations. Just like Agile, DevOps originated from daily practice and was a way to get Ops more involved in development and testing of software. It's related to many trends in Software Development which were (and still are) booming when the term was first used in 2009: e.g. Lean, ITSM, Continuous Delivery/-Integration, etcetera. So, we can say goodbye to those expensive (and often annoyingly critical) testers! Hurray! Thanks to Operations we learn from production. Developers can build some nice automated regression tests which are part of the Continuous Development/-Integration pipeline. Quality Assurance is dead!
Right?

Truth be said, some companies actually did say goodbye to testers. And they're still standing. They claim that developers are best at automating tests, they know best what software they've been building and why. So they are perfectly capable of testing their own work, together with the end user (representative). Indeed with DevOps there's a strong focus on test automation. Like Agile the work pace is high: Software is developed iteratively during short sprints, so regression tests need to be running all the time. This can only be achieved by using test automation. So, maybe developers are indeed the best testers (test automators) in this 'new and improved' Formula One pit street?

However, like Elisabeth Hendrickson claimed at the OnAgile Conference in 2015[5], test automation can only be used to check if software meets a certain expectation. *And checking is not testing!*[6] This means that Test Automation will most certainly not replace the QA specialist. Mainly because there are two things at which humans still (massively) outperform computers and those are learning and creativity. Both important aspects used in exploratory testing. Therefore test automation should mainly be used to free up time for the software tester to focus on testing using exploration and experimentation. So maybe the earlier mentioned old school software tester (expensive, annoyingly critical) is indeed dying (or actually quite often already dead and buried...). But the Quality specialist who helps create a creative, quality focused mindset based on continuous learning to the entire DevOps team is very much alive and kicking!

So, where is QA in DevOps? It's very much at the center of it. The software tester has always been right between Dev and Ops. Trying to slow Developers a bit down when too eager at change and innovation. And at the same time putting some effort in taking Operations along in the inevitable change that's on its way to them. Within DevOps this role is probably more important than ever before! The Quality specialist (or Quality Analyst, Quality Engineer or even Quality Ambassador) is in the driver seat and uses his/her creativity and

[3] https://legacy.devopsdays.org/events/2009-ghent/ (2017-01-23)
[4] http://www.forbes.com/sites/michaeldefranco/2014/03/04/not-eating-your-own-dog-food-you-probably-should-be-2/ (2017-01-23)
[5] https://www.thoughtworks.com/insights/blog/qa-dead (2016-12-22)
[6] http://www.developsense.com/blog/2009/08/testing-vs-checking/ (2017-01-23)

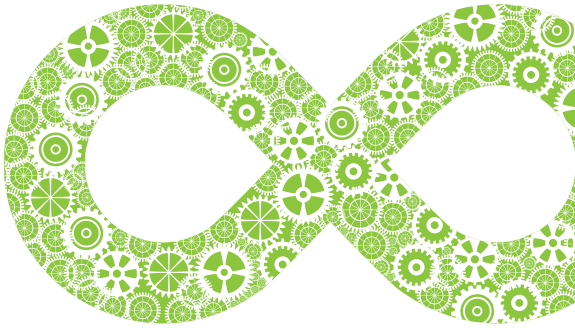
↗
Berry Kersten,
Berry is an experienced software tester. His passion for quality and agile software development is reflected in several publications and presentations. Regarding DevOps, Berry is a certified DevOps Master and he shares his knowledge at conferences like the upcoming DevOpsDays in Amsterdam and in Ghent last year.

You can follow Berry Linkedin: https://nl.linkedin.com/in/berrykersten and contact him at Berry.Kersten@ImproveQS.nl

ability to learn to set a certain pace for the entire team in which software is released as soon as possible while maintaining a certain level of quality. This can be done by keeping an eye on the big picture. Decide what tests need to be automated and when it's better to test manually. It can be decided to execute tests manually because risks are low and tests only need to be done once (or every once in a while). However it's also possible that an experienced tester expects to get more out of manual testing compared to automatic checking of software, which could be useful when risks are actually high! This is where learning and creativity plays a central part. One of the biggest weapons this quality ambassador has to take advantage of this human touch to QA is Exploratory Testing. This can ultimately give insight if delivered software is actually of sufficient quality. It's basically a mean to set the speed limit at which the (software development-) car drives as fast as humanly and technically possible while staying on the road and not falling apart in the process.

(Exploratory) testing in DevOps

So, what is exploratory testing? To quote James Bach[7]:
'The plainest definition of exploratory testing is test design and test execution at the same time. This is the opposite of scripted testing (predefined test procedures, whether manual or automated). Exploratory tests, unlike scripted tests, are not defined in advance and carried out precisely according to plan.'
This is exactly the form of testing that focusses on the two earlier mentioned things in which we humans still excel in comparison to computers: learning and creativity. A computer, and thus an automated test, cannot perform exploratory testing like humans do. And that's the main reason why exploratory testing is so extremely important within any software development project, but even more within DevOps projects where test automation is a necessity, however no silver bullet.



While all team members, Ops included, tend to more and more focus on speed, speed and just speed. It's up to the tester to also keep a focus on learning and creativity in order to keep a balance between speed and quality. QA is the accelerometer for the entire DevOps team that helps make software development as fast as possible while maintaining a certain level of required quality.

Does this mean every Software Tester should start specializing in Exploratory Testing? Is Exploratory Testing the last thread that keeps QA alive? I believe it is and like James Bach and Michael Bolton claimed before, I'd like to point out that software testing in general is all about exploration and experimentation[8]. Or at least, it should be! Therefore, exploratory testing is essential to keep delivering high quality software as fast as possible. Especially when dealing with a DevOps environment which may be considered the next logical (and maybe even inevitable) step within software development. So, testing through exploration and experimentation is (one of) the ultimate means to put QA (back) in DevOps. It can help prove that QA might even be more important than ever before within this DevOps environment. Not per se the test specialist as a person, but more the general mindset of the entire DevOps team. A mindset in which quality, learning and creativity plays an important role and where 'quality thinking' becomes part of the teams DNA. That's where QA is in DevOps.

The authors would like to acknowledge their colleagues Huib Schoots and Joris Meerts for their contribution to this article.

7 http://www.satisfice.com/articles/what_is_et.shtml (2016-12-27)
8 http://www.satisfice.com/blog/archives/856 (2017-01-23)