[00:00:03] **Don Jackson** Hello and welcome to our session today to discuss our point of view around the shift culture that's permeating the software testing industry. I hope you're enjoying the event as much as I am. My name is Don Jackson and I'm a chief technologist here at MicroFocus.

[00:00:16] So you may have heard of shift left or shift right being essential to successfully preventing defects slippage in today's application development landscape. In fact, the 2021-2022 World Quality Report cites that 55% of respondents stated that they needed to shift left more than 53% of those same respondents stated that they needed to shift right more.

[00:00:38] Our session today is going to focus on the MicroFocus point of view that we need to shift everywhere both left and right, but not abandon the lessons that we've learned so far in testing. Well, before we jump in, I wanted to mention that the MicroFocus universe event is coming up in a little over a month. You can scan the QR code on the screen and get registered quickly and simply. We'll be talking a lot about Shift Everywhere, Value Stream Management, and digital transformations during this year's event. With that being said, let's talk about Shift Everywhere. So really gone are the days of being able to toss the application over the fence to QA when you finish coding.

[00:01:15] The speed of business is too fast to allow for the delays introduced through the feedback being sent days to weeks after the developer finished the code. The human mind has such a challenge context switching from what is currently in the forefront to what was being worked on that long ago. That delay is further compounded by having to ramp back up on what the developer did prior to the commit. We in I.T. have provided the solution to this, and that's delivering software to a DevOps pipeline. But what is a DevOps pipeline?

[00:01:46] A DevOps pipeline can be visualized through a water pipeline. There are gates represented by the valves at different points and if a failure occurs, represented by the production feedback at the bottom. Feedback is sent back to the submitter, usually a developer, immediately to reduce the likelihood of contact switching delays. The submitter then resolves the failure, then resubmit it into the pipeline when the submission occurs, it should be stored in the source code repository. At source code repository should version the changes, including submitter comments, provide collision detection, which is if multiple changes against the same source need to be integrated from multiple submitters and merging capabilities.

[00:02:26] Once the source has no collisions, the build system will take over and start executing the pipeline steps. This is called Continuous Integration. The first step is to compile the source into to create artifacts like, for example, a jar file or war file for Java code and then to store those artifacts in the artifact repository. Once the artifacts are copied to the artifact repository, which provides a central location for all artifacts, the build system then executes unit tests in the testing system.

[00:02:57] This is the first step of continuous testing. Assuming that the unit tests pass. The build system then engages the provisioning and deployment solutions to stand up the next preproduction environment, then deploys the artifacts to that environment. That's called continuous delivery. Once the environment is up and running, the new artifacts. The testing system then executes the appropriate automated tests for that environment.

[00:03:21] This is another part of continuous testing. The build system then has the provisioning system, the provision the environment, those last three steps are repeated for each environment or type of testing that needs to be done could be integration, system testing, performance testing, whatever. Once all the testing is completed successfully, the build system has the deployment system move the artifacts into production, including any steps necessary to have the application start to leverage those new artifacts. And that might be things like JVM restart or a full system restart, maybe just cache cleanup or something like that. And that's called continuous deployment. As part of this deployment, the operations and monitoring artifacts, which could be scrips, configurations, alerts, other things like that are deployed and activated.

[00:04:10] These should have been also included in the lower environments, and that's called continuous monitoring. So now that we set the stage of what is a DevOps pipeline, let's drill into testing, which is what we're here to talk about. So we talked earlier about the reason why we can't just throw stuff over the fence to QA, but that doesn't mean that the QA persona is no longer needed. The QA person provides a critical and different way of thinking how to test the solution. QA personas QA people usually have a destructive mindset.

[00:04:42] They're the resource will come up with and execute, hopefully in an automated way. The majority of the negative test scenarios or the artifact. A good example of this might be a quantity field on a shopping application. A positive test which will be handled in unit testing would be to test the boundaries of the field. So let's say it's a two digit field. The data scenarios covered by the positive tests would be the lowest, the number one, the highest number, ninety-nine in some random value in the middle. And this would typically satisfy the unit Test. A QA person, though, would come up with negative testing scenarios.

[00:05:16] So what happens if I type in zero or negative one right paste in 99 billion or a pound sign, or some Cyrillic symbol or an emoji or some hashtag or something like that? While QA provides this necessary and valuable insight, there are definite limitations to what QA can do. Normally, QA is doing black-box testing, meaning that they have no visibility into the inner workings of the artifact and thus can't ensure proper code coverage. Further, QA is typically engaged, as we discussed earlier later in the cycle, increasing the time for failure feedback. This is why we need to include the Dev Tester or the SDET Software Development Engineering Test or developer into testing. The Dev Tester should be focusing on unit-level testing, and because they're the ones typically writing the code will be doing white box testing, meaning that they can validate the inner workings of the artifact. All of those logic permutations and make sure that their test has proper code coverage.

[00:06:19] They have immediate feedback as well, so there are no delay or context switching problems like we talked about earlier. Now, this graphic can help you visualize the waste shift left and shift right or viewed with the developer testing being as far left as possible. You can also visualize why it's called shift left by thinking about the positioning of the activities and a Gantt chart in a project plan, with the development activities happening much earlier or left in the Gantt chart. While the developer understands the code, it can ensure adequate code coverage and get the fastest feedback by creating and executing tests themselves neither the development nor QA has intimate knowledge of how to run the business utilizing the artifact.

[00:07:00] This is why it's crucial to also shift right your testing by engaging the business analysts or domain experts into the testing, and they should be contributing to the test automation. This persona is the person or people who understand the business processes why the artifact is being built or modified and how that artifact will be leveraged to run the business. They know the flow through the business and they understand what is critical to keeping the business going and what workflow will be executed through the solution. Utilizing the artifact in question, this persona typically doesn't know how to code and can even be scared off from doing work if shown code.

[00:07:39] He might run screaming for the hills. It's critical that their contribution towards automation doesn't require coding skills, yet still can have those contributions be automated. When you think of this persona, you can think of a banker at a bank. Someone that has little to no technical knowledge, but their contribution to the business knowledge into the test automation is crucial to getting as close to zero defect slippage as possible, which should be our goal.

[00:08:08] We at MicroFocus provide solutions for all three of these personas. In the functional test automation space for the QA persona, we provide a solution called UFT one. This solution is infused with our AI model, providing computer vision and OCR capabilities to the QA persona, enabling faster scripting and more resilient tests than properties-based scripting. Essentially, computer vision identifies the objects on the screen the same way a human does by their visual characteristics. Now, this screenshot shows the AI inspection window when a computer vision has identified multiple objects and text blocks on the screen that can be acted against.

[00:08:46] But for the developer persona, we provide UFT developer. UFT developer has also infused with the AI model, but is delivered via a library so that in the developers' IDE. In this case, we're displaying IntelliJ. But since there are libraries, they can be imported and leveraged into any IDE that supports the language that you want to script against Java, JavaScript, or C#. If you're using Visual Studio, Eclipse, IntelliJ, the Spring Tool Suite, JBoss Developer Studio, CodeReady, Developer Studio, or Android Studio, they also get the benefit of the UFT developer menu and their associated wizards to help the developers be even more efficient in their test creation and maintenance. UFT Developer also includes a local version of service virtualization at no additional cost, so the developers can leverage that and enable them to test and develop their artifacts when other components of the solution aren't available.

[00:09:42] So if you think about something like a pricing and external pricing engine that you have to pay per use, you can use service virtualization to simulate that when you're doing your development and your testing. For the business analysts, we provide a product, a new product called UFT codeless, which is truly codeless as opposed to code suppressed like some of the other solutions on the market. And this leverage is natural language processing or NLP, which comes with our newly released model-based testing solution. As you can see, the quote unquote script is near clean, plain English.

[00:10:20] No business analyst should be scared of this step that says click the profile or click the speaker's text or type the username value into the username field. This also leverages that same AI model. So when we talk about the AI model, we're talking about some components, and I thought it might help if we drill into it a little bit. The first one is computer vision. So we talked a little bit about this, but it identifies the same way a human does without using those underlying identifiers.

[00:10:49] If you think about something like Selenium where you're dealing with frequently the XPath, it doesn't use that. It uses something. It uses those parameters on the screen, the visual characteristics, and clues on the screen. So a shopping cart icon might look like half of a hexagon with a line underneath it representing the shelf under the basket, a couple of docs underneath that representing the wheels may be a squiggly line to the side that represents the handle.

[00:11:18] These are all visual characteristics instead of the underlying identifiers, which means you're not reliant on what the operating system exposes to you. It doesn't matter what the underlying technology is that built it, whether that was one development framework or another. It's what's visually representing.

[00:11:41] Then we have a neural network in there as well. So this is how the system will understand how to interact with it. So if you think about like a magnifying glass for a search icon, well, the way that you interact with that, if I told you that search for something is you click the magnifying glass. You'd expect a search box to come up that you type in the search value and then you click the magnifying glass again to run the search.

[00:12:05] That's how you natively interact with it. And then, there are the NLP capabilities that we talked about whereas you saw, the tests are written in near plain English. And that's going to continue to evolve to do eventually, we'll be able to handle nuance, a bit of the language as well. Just like in the functional test automation space, though, we also cater to all three personas in the performance testing space. For QA or performance engineer, we provide LoadRunner, a professional for smaller project-based initiatives. When the initiative is larger or enterprise-wide, we provide LoadRunner Enterprise, which extends the LoadRunner professional capabilities by including asset sharing and trend reporting.

[00:12:47] The asset sharing allows you to share lab resources, so the load generators, the controllers, your licenses, and schedule and reserve them. Trend reporting provides a mechanism to analyze not just the current run, but how that run compares the previous runs to provide the larger picture of whether overall performance is improving or degrading from run to run. We also provide LoadRunner Cloud, where our SAS provides all the necessary infrastructure for executing the tests at runtime in our cloud. For the developer, we have a relatively new solution called LoadRunner Developer.

[00:13:19] It's free. It can run up to 50 virtual users within the developer's IDE. It can also run in the LoadRunner Cloud as well directly from the IDE and can execute in conjunction with the UFT developer so that the developer can understand the impact of injecting load on the actual end-user. Additionally, since UFT developer and LoadRunner developer reside on that in the IDE, you can leverage the service virtualization capabilities during execution as well to enable performance and load testing even when other parts of the solution aren't available. As you can see from the screenshot, we also leverage Grafana Analysis Dashboard during the run as well, including with LoadRunner developer.

[00:14:04] For the business analysts, we have a protocol called TruClient that allows the business persona to contribute to performance testing as well. As you can see, the script is displayed in near plain English again and allows most of the business analysts to create scripts that are 80 to 95% complete in delivering those scripts for final polish to the QA or PE persona. Everyone should be contributing to performance as well.

[00:14:27] Now, let's drill back out to look at the DevOps pipeline. You probably read multiple articles about DevOps from multiple sources, including consulting firms. Most of those articles extoll the virtues of open source and suggest you should definitely leverage open source solutions since they're quote-unquote free. Now, each box has multiple solutions that are viewed as open source. The challenge that's being faced in the industry is that these quote-unquote free is only talking about the licensing costs, not the total cost of ownership. These quote-unquote free solutions frequently require significant human labor investment in order to successfully accomplish the test or the test that tools being asked to do.

[00:15:06] So let's take it Selenium for example, it requires detailed coding knowledge, intimate understanding of the object properties within the DOM or the document object model of your application, and only works with web locators often rely on CSS or XPath. Now, many technologies many frameworks will cause the CSS and or the XPath to change with each build of the artifact causing your scripts, your test automation scripts to be brittle and break. Furthermore, the scripts are so technical that they're difficult to read and really only cater to one of those three personas that we've talked about. As with all Open-Source solutions, when not if you run into a problem, you have to rely on the open-source community for a resolution.

[00:15:47] If the problem is unique to your environment, you're on your own to figure out how to resolve the problem. If the community hasn't encountered your technology before, you're also on your own. When you deal with a COTS or commercially available off-the-shelf solution. Like MicroFocus Solutions, you're backed not only by a robust user community similar to opensource but a supporting R&D organization who can help you when you face your challenges, providing you a much faster mean time to resolution and deeper and broader technology support. As with the case of Selenium, since you're limiting the personas that you can contribute to automation, you're also exposing yourself to the increased risk of defects slippage because you haven't accounted for those other personas and their processes and their thorough knowledge.

[00:16:33] Let's assume that you've done all of that necessary human labor investment, and your pipeline is running smoothly, efficiently, and effectively utilizing all of these open-source tools. But how do you make informed decisions about what's going on in that pipeline? So we at MicroFocus, we provide solutions for each of these phases of the DevOps pipeline. Well, we wouldn't be opposed to a rip and replace approach to your pipeline, we really focus on plugging the gaps in your pipeline as they exist today while augmenting the return on your investment where you don't have gaps. For example, if you tell us you already have a team of Jenkins experts who have built and maintained Jenkins jobs and pipelines and the build system.

[00:17:12] That's great, continue to do so. And we can plug Jenkins into octane to increase the value. Maybe you need to plug the Git repository gap that allows developers to overwrite previous developers' comments without a history. We could lay pulse soon on top of Git to provide that immutable history while preserving the current way of working. Or maybe you aren't performing static code analysis, and we can help plug that gap with Fortify. Or maybe you're using your build system to also do deployment with lots of scripting, which requires lots of maintenance. We could help you by digitizing those scripts with deployment automation and stop forcing your build system to do something it wasn't purpose-built to do.

[00:17:53] But there's still a challenge that we need to address. There's actually a fourth persona that we haven't talked about yet, and that persona is your leadership or your management. They need to be able to make informed investment and strategic decisions based on the information in the entire pipeline across all of the different tooling. You're not going to have a director or vice president scraping through Jenkins logs, trying to determine what's going on within the pipeline. And that's where ALM Octane comes in. ALM Octane gathers the information from across all of your DevOps pipeline tools, whether they're MicroFocus open source or third party, and provides reporting and dashboarding so that your leadership can make insight-driven business decisions.

[00:18:37] Driving your DevOps pipeline to the achievement of its goals. Delivering software quality at the speed of business with quality. And then that will fall into your value stream management so you can start measuring value, which we'll talk more about in future times.

[00:18:54] Thank you for taking the time today to talk with us. I've got a hyperlink and a QR code to an e-book that talks about integrating open source and cots solutions and the value you can realize by doing so. Thanks again and enjoy the rest of the event.