EPISODE 1371

[INTRODUCTION]

[00:00:00] KP: It wasn't that long ago that many companies scheduled downtime in order to release an updated version of the software that runs their website. That's rare today. Most developers want continuous testing, integration and deployment continuous everything. With that comes many benefits. It also places greater demands on quality engineers who can no longer gate all updates into a single infrequent release. Liliya Frye is the director of QA engineering at LeagueApps, a provider of sports league and team management software. In this episode, we discuss Liliya's experience and modern practices for successful enterprise strategies.

[INTERVIEW]

[00:00:43] KP: Liliya, welcome to Software Engineering Daily.

[00:00:46] LF: Hi, Kyle. Thank you.

[00:00:49] KP: To kick things off, can you tell me a little bit about your journey as a technologist?

[00:00:49] LF: I'd say I've been in IT for the last 24 years in web development roles, testing engineering, managerial roles and different leadership positions. I specialize in creating quality engineering processes with continuous testing, shift left, and shift right methodologies with continuous monitoring, continuous improvement in scaled agile software development life cycles with strong focus on customers, teamwork, and organizational overall success.

[00:01:29] KP: Could you share a few details on what you mean by shift left and shift?

[00:01:33] LF: So, shift left and shift right, one of the methodologies that specifically shift left testing was introduced by Larry Smith in 2001. Now, it's been 20 years ago, where he mentioned that bugs are cheap when caught young. He introduced the mindset where the quality in testing should start early in software development lifecycle to reduce the number of bugs and the quality ownership, means better requirements, better design, better code and better test. With shift left, he also mentioned that the quality is owned not only by the quality engineering team or quality assurance team, but it's owned by the entire engineering team, including product managers.

With shift left, the quality engineering team members, there are part of the agile process. And they're embedded with the scrum teams with the squads. With shift left, we continuously thinking about how can we test throughout the multiple stages in software development lifecycle. And shift right, it's method of testing in monitoring in production. This approach helps developers to uncover new unexpected scenarios, that could not be detected in lower environments. And their way we can fix bugs before the users find it.

However, shift left, it's specifically for – usually we test in production manually. In automation, you have to be very cognizant in which area of application you introduce with shift right methodology.

[00:03:54] KP: You'd also mentioned one of my favorite buzzwords, and that's continuous and a couple of areas of interest and areas of expertise you have, whether it's something continuous, or maybe a different philosophy or project, are there any standout improvements or schools of thought that have come around in your career that are starting to really shape the way people build software?

[00:04:15] LF: The continuous, I see it as we are not – if we set up processes, and especially for quality engineering, you create strategy, you set processes, the processes that you set up, it's never going to be written in stone. You have to continuously think how can it be better? How can we improve it? And improve it with like a different QA strategy, different environment strategy, execution strategy, data strategy? With automation, how can our automation in different areas in implementing the test-driven development, unit testing, integration testing, API, UI end to end UHC. We're constantly thinking how we can make, create a certain flow, that with continuous integration, continuous delivery and deployment, that we find bugs and kind of fix them very early very smoothly, and they do not affect our end users.

[00:05:35] KP: Well, tell me a little bit about your current role today, and what led you to take an interest in that area?

[00:05:41] LF: Currently, I'm working at a LeagueApps, and LeagueApps is a fantastic company. We provide an operating system for youth and local sports leaders. We have an application that could be used on desktop web and also mobile application for iOS, Android. We support about 3,000 organizations, and our organization can also create through us their customizable website, which is branded. Basically, we provide the tools where you can create your own website that is presenting your organization, sports organization, and we provide you with the tools where the branded apps could be created for iOS and Android devices.

Mobile applications, specifically, help you with registration –not registration with keeping the schedule, make sure you're understanding what's happening, when are the games going to take place, when are the tournaments and you can communicate through our mobile application with your teammates, with your coach, with organizers. Let's say often, certain games are being canceled due to some weather or like emergency. So, then the typical push notifications are sent and people either parents or players can receive it very quickly. We're also working on a way where our mobile application would help players to engage more. We’re going to offer more features where they would learn more things about the particular sport they're playing, or they can engage more with each other.

So, there's a great kind of service we provide for sports organizations. And my responsibility is, I'm a Director of Quality Assurance Engineering, and I want to make sure that the quality of application is up to standards and our end users are happy and they can assist their players with all the necessary tools that they can just focus on doing their best, playing the game, and know when everything's happening on their schedule.

[00:08:25] KP: What's the team look like in terms of current size and growth plan?

[00:08:31] LF: Currently, our team is, I wouldn't say huge. We're very limited. But what's so unique about LeagueApps, we take the entire organization and our company is responsible for quality. And that's what I like about LeagueApps, that everybody, we're talking about the customer success team, the product management team, the developers and QA team. We're all thinking about testing, and everybody takes ownership and responsibility for quality. It's not like in certain organizations where if a bug appeared in production, and everybody is pointing fingers at the QA team, no, it's not like that.

Here, if something happens, if some issue occurs in production, we all collaborate and work together and do the risk assessment and mitigation plan to figure out how can we prevent it from happening in the future and the growth, speaking of growing the team, since we're transitioning to migrating our old system into the new, the next generation, we are going to certain areas of application, going to be migrated into micro services. Therefore, we'll need lots of testing. Currently, my team specifically looking for a mobile aesthete, and also senior software development engineer and test, and the senior aesthete will be focusing on backend testing, API testing, and anything backend related and mobile listed, anything related to mobile applications, native, iOS, Android, automating. The test cases for mobile native and mobile web. Also, we're looking for a lots of engineering roles. We need front end engineers, platform engineers, senior backend engineer, senior senior front end, Android engineers, and we're looking for VP of engineering, who will help us to move to that next gen, with microservices architecture.

[00:11:02] KP: Very cool. Well, a lot opportunity there for many of the software engineers listening to this podcast. Where can they learn more?

[00:11:09] LF: They should go to our website, which is leagueapps.com, and under career, there should be, I think it's under company career. Yeah. Or there's another option to enter into browser, careers.leagueapps.com.

[00:11:35] KP: Very cool. Well, I'm wondering if we could zoom in on the roles that you're hoping to fill in test. What is it that you're specifically looking for, that people might want to beef up on? Or should have be an expert in already? What are the skills you need to add to your team?

[00:11:49] LF: So for my team, specifically, for a senior software development engineering test, we need somebody to test our API's that are built in Java, and somebody who is comfortable with JUnit, Mockito libraries, and the who had worked with Postman, and who can help us with building CI/CD. We're in the process of taking all our automation scripts into the CI/CD pipeline, and we need assistance with that. Somebody who integrated automation scripts with CI/CD, and who can help us to create the continuous integration delivery deployment, specifically, with understanding how are we going to trigger our automation scripts in one environment and how they're going to be moving from, let's say, from dev environment to QA, and to pre prod, and prod. Somebody who has in depth understanding of CI/CD integration.

We are looking to – we're exploring different CI/CD tools. If somebody has expert in working with different CI/CD, may be Travis CI, GitHub auctions, and the majority of our applications are in Google Cloud, GCP and we have some in AWS. If somebody has knowledge of GCP, that would be great. And for mobile listed, we need somebody who has experience with Appium. And with Appium, specifically, we want somebody who knows how to build applications that build one framework, that could test immediately, the native applications, and the native application iOS, Android, and the mobile web at the same time with one framework. The language, that could be either Java or JavaScript. And also, the knowledge of CI/CD would be helpful.

[00:14:19] KP: I have used the headless browser Selenium to do some tests like front end test automation, in a couple of cases, not really my area of expertise. Is there a Selenium for mobile?

[00:14:32] LF: The Selenium for mobile would be Appium, because Appium is based on Selenium, and that's what we're going to use. Our front-end applications, I mean, the front-end automation scripts are in Cypress. Originally, we had in Selenium. However, we had so many flaky tests and we had so many issues and we transitioned into using Cypress We also will be looking for front-end automation engineers to build a lot of lots of end to end test in Cypress. But currently, there is no budget for that particular row, which will open up in January.

[00:15:19] KP: I've seen a lot of different approaches to where the line is drawn between a software engineer and a quality assurance engineer. Seems like every company has maybe a different take, and that's probably okay. But it also can be a place where there's conflict, if it isn't well structured. You could have a resentful QA person who feels the software engineer is being careless because they know that person is there to take care of them later or something like that. Do you have any philosophy or thoughts on how to draw the line between those two roles and know whose responsibility is whose?

[00:15:51] LF: So, you say, between software engineer and QA engineer?

[00:15:55] KP: Yeah.

[00:15:56] LF: Okay. I would say, the software engineers, they are more responsible for building test driven development. And also, for unit tests and some integration tests. Quality engineers, we build more like a functional unit test. Software developers would build like a sanity smoke unit test, very small. We're focused on more in-depth unit tests, and also integration tests. We’re responsible for API tests. Again, more in-depth suite and we're responsible for UI, anything front end testing, and building end to end test. UAT, user acceptance test, it depends on our organization. Some organizations, their QA responsible. Others, they have a different department. In our organization, we have product managers, who are responsible kind of with the engineering managers.

[00:17:15] KP: Make sense.

[00:17:16] LF: Yeah. In performance testing, we are actually sharing the performance testing responsibilities between QA engineers, and the developers, because there are certain performance tests could be done by developers. For example, they just want to make sure that they identify and test upper bounds, lower bounds, and they want to make sure that pagination is implemented. Our QA team, mostly doing some load stress and scalability testing.

[00:17:58] KP: Performance testing can be especially challenging in my experience, because even the best of everyone's intentions, we can't predict all the crazy things the users will do in the wild, especially if we have a very widely used application. There's going to be edge cases we haven't thought of. How do you approach the unknowns in a situation like that?

[00:18:20] LF: Usually, I try to create a plan where we identify a certain month of the year when we expect big concurrency of users. Usually, in our business, it's in the summer where we have tournaments take place and you try to create a plan where you do a stress and scalability test with multiple concurrence at the same time. The tools to use for performance testing, that could be either Gatling or could be JMeter, BlazeMeter. With performance testing, usually if you do some stress and scalability testing, you try to kind of increase the load request until your environment fails over, and with that particular stress testing.

For example, one thing I want to mention, this is more like on a personal observation. When the school started in mid-August this year in LAUSD, my son is a senior in high school here. On the first day, each student is supposed to present a daily pass. And on the first day, there was 400,000 students had to present the daily pass. What had happened, LAUSD did not do sufficient performance stress scalability and chaos engineering testing, their service failed, and no one had access to their website. No student was able to produce a daily pass. Even those students who supposed to present a result for COVID test, if they had negative, those results were in their website, and they could not access that website either to show them if the result is negative to access on the premise of school.

It created a huge chaos on that first day, for about the students were standing outside for an hour or two, were not able to enter the school. So, for me, performance testing is extremely important. Especially when you know, on a certain day or a certain month, you're expecting huge concurrency of end users.

[00:21:01] KP: Well, a lot of continuous tools, continuous CI/CD, continuous testing, all these good things, it seems to me, there's a pretty strong agreement, everyone knows that company should move in these directions, this is the right way to go, it's the efficient path. Yet for some reason, anecdotally, I observe not everyone is quite there yet on the growth to a mature ecosystem. If you have to maybe convince someone that resources should be allocated or time should be spent really modernizing an infrastructure like this, what are some of the low hanging fruit or early gains that the company can expect when they place an investment in these areas?

[00:21:39] LF: So, I would say the most important area to invest is in deciding, first of all, which CI/CD tool to use, and deciding on what area of application would benefit from automation coverage. I believe, to improve the quality, you need a lots of automation, but you cannot automate everything you have to use, I would suggest to use Pareto Principle with the 80/20 rule that you focus on automating the area that is the most used by your customers or end users. And your do your best to cover as much as area of application as many kinds of the different paths or different flow and take those automation scripts to CI/CD. There are deployed the automation scripts that are, you know, automation tests, they are triggered automatically as a part of the build pipeline.

Let's say, if the build is tested in one environment, then it propagates to another environment and another sets of automation script are triggered. And then after going to another environment, so you create this flow of continuous deployment and depends on the organization. Certain organizations are deploying to production automatically if it's a safe environment. I believe it's very important. Currently, to focus on continuous integration, continuous delivery in deployment, and look for talents that have experience with utilization in integration with CI/CD tools.

[00:23:53] KP: You'd mentioned step one being to pick a CI/CD, I don't know if you said platform. If I just impose that on here. Do you have any opinionated recommendations about the primary tools people should be considering?

[00:24:05] LF: Okay, I would say you have to know Jenkins for sure. Speaking of tools, there is debate, there is pluses and minuses with different tools. I would say, I cannot recommend for sure. GitHub actions, Travis CI. I had worked with Circle CI and GitLab Ci, but we had certain issues there. But it might work for other organizations. I would say it depends on organization what their specific needs. I cannot advocate for a particular tool, per se.

[00:24:49] KP: Totally makes sense, yeah. Well, Jenkins is a mature and established technology. It's been around, I think everyone knows the brand name. It's almost surprising that it hasn't been knocked off the king of the hill position in some sense. What about Jenkins makes it still relevant today?

[00:25:06] LF: You're right. Jenkins has been around for 10 years. And Jenkins, the benefit of it, it's open source and it's still relevant. The biggest benefit of Jenkins, lots of tools can integrate with Jenkins and we're talking about like Jira, different test management tools can integrate with it and different cloud service applications can integrate with a different cloud, like a pipeline. So, Jenkins also works more of a glue, even let's say you're going to utilize some other CI/CD tool, there are certain areas where Jenkins still works as glue connecting different parts together. I don't see Jenkins is going away. It's been great tool for one decade, and I see it's going to continue.

[00:26:09] KP: What do you think the general quality infrastructure stack looks like in the future? Are we just going to kind of continue on the path we're going on? Are there innovations that you think are going to come at play that make it easier in the future?

[00:26:23] LF: In the future, I see more of, we're going to use AI driven tools, which currently, there's certain companies provide services where you can write front end, end to end testing, with some AI, ML applications that can recognize certain flow that end user uses. Also, they have certain OCR algorithm to scan the page, and they build, they train their machine learning models, to kind of build automation scripts. Also, lots of cell healing tests would help with front end testing and some natural language automation.

What would really help, honestly, currently, there's just a limited number of companies that are offering that, but you have to pay for that service, not open source, and what would help in the future, to have certain open source tools that would offer that, because currently, the challenges is for a QA engineering team, to set up automation, to build a framework, have adequate skills on the team, and where are we going to run it, all the technical details, and also deciding what type of scenarios we should actually test just designing the test cases. If that particular area would be covered by AI/ML tool, it will really speed up the process. I really wish in the future where this kind of tools would be open source, so everybody can use it. If that would happen, then it would really put us on a next level of efficiency and checking the quality of our applications. I strongly believe it will improve the quality of web and mobile applications around the world for everybody.

[00:28:47] KP: What's an exciting vision as that starts to happen? And machine learning and AI are doing more of like, let's say the one second or low-level work that frees up a quality engineer to, as they say, stand on the shoulders of giants, how do you see it changing the current role and what professionals of the future will be doing?

[00:29:05] LF: I see that quality engineers will have to build a relationship with – it’s it's going to be sort of test op, ML ops, working together. And quality engineers might have to pick up some understanding of how to train machine learning models. If they want to sort of become more in demand in their organization, not only quality engineers currently, before they had to know how to write automation scripts for, let's say, unit API front end. And then they had to learn how to use continuous integration tools. Now, we're talking about they have to understand the machine learning technology and how to train models. So, that's kind of the future skills they have to obtain.

[00:30:08] KP: So, for a growing company, maybe one that's growing a little bit too fast on their successful product and the software side, and it's time to really make that investment in good quality pipelines and CI/CD and things like that. Is it useful to think of that as a onetime reaching some milestone and then being in kind of a maintenance mode? Can we say we're going to clean up all the technical debt, build all the CI/CD, and we're done? Or what does that realistically look like in practice?

[00:30:37] LF: Oh, you can never be done when you build CI/CD. Unfortunately, there is always going to be issues with test data or test environment. And with even CI/CD pipeline, you have to maintain it. The only thing what would improve is how fast you find bugs and how fast you fix it. And yet, with continuous integration, delivery and deployment, you have to maintain it. I would say, the only thing what could improve is, especially if we introduce an AI/ML, then our quality engineering team will be focusing on learning how to maintain it. That will kind of transition into the maintenance roles.

[00:31:37] KP: So, when thinking about some of the data that's going to flow through the system and the test data cases, you'd mentioned. Obviously, you can make up your own test cases as you go with any data, but it's not necessarily going to represent what it looks like in production. Companies are often protective of their production databases for good reason. I'm curious if you have any thoughts on how a quality engineering team, what relationship they have with real production data? Is it reasonable to get a batch of it and run tests? Or does it need to be anonymized? What are your thoughts there?

[00:32:12] LF: So, with testing and production, yeah, depends on the organization and what kind of data is flowing. I would suggest, yeah, there should be the careful consideration for testing and production and what exactly should be tested if some structural, or functional database. And I would say that, it should be the limited number of data flowing. The safest bet is to test any data quality in a lower environment.

[00:33:03] KP: So, when you have a nice CI/CD environment setup, if everything's working perfect, it's going to deliver great software. When a mistake is made or a bug is introduced somewhere, it should fail, right? It should find that error. And then something needs to happen. There's an operational procedure that maybe it's a little bit beyond the tool itself. Do you spend a lot of time thinking about how to triage?

[00:33:26] LF: Yes, we do. We also think about the rollback process, and especially talking about planning for the failure and making sure that our application is decoupled and are selecting the right technology. Especially, when rolling back, we are checking what exactly – how fast could it be fixed and what area. That is something we're constantly working on improving that process. Again, it's a continuous improvement and you learn on your mistakes, what had worked in the past. The goal is honestly, if you find certain issues in specially some critical issues, make sure you fix it very fast that end users will never find out.

[00:34:33] KP: When I think about doing a rollback, in my mind, it seems easier if I have some monolith. Yet there's such a trend towards micro service architecture. It feels like it might be harder to rollback there. I'm curious if that's the case, or if there's anything else about the popular micro service design that makes your job extra difficult?

[00:34:53] LF: I would say there is a certain area that with micro services, there is some dependency of one application into another. And we consider it when we build those applications to reduce that dependency, so that we can manage the rollback of that particular component, and that would not affect the sort of the other areas. Usually, we just diagnose it thoroughly to mitigate any – just to minimize any risk.

[00:35:35] KP: Make sense. Yeah. Similar question. I'm thinking about tools like Docker and Serverless, popular ideas that have come about somewhat more recently. And these are new challenges, you have to figure out how to do testing and build pipelines around them. Are there anything specific about containers or Serverless, that are novel or interesting from your point of view?

[00:35:55] LF: Speaking of Docker, it was introduced 10 years ago by Solomon Hykes. Basically, at the same time as Jenkins. We utilize Docker very heavily. It really helps us to use the pack shape and run any applications and virtually anywhere. With Docker, when the containers are created inside a virtual machine, they really provide this ultra-portable solution. Sometimes we can spin up some transient environments as needed. And of course, the Kubernetes that was created by three people, Craig McLuckie, Joe Beda, and Brendan Burns in 2014. It really helped with container orchestration, to scale CI/CD pipeline, and where you can deploy containers in the cloud and schedule batch jobs and handle workloads and easily perform rollouts, it makes the whole process more efficient. That's when Kubernetes become handy. At our current company, we do not utilize Kubernetes yet, but we're working towards it. But that's one of the tools that I'd say every company should utilize.

[00:37:28] KP: We're certainly moving towards continuous everything and a lot of ways, I think for some good reason. And obviously, just from our discussion, I can imagine you're a big supporter of that. I'm curious if there's any pessimistic side to it. Does the rate at which software is changing and the tools we're making to allow it to change faster, is that intimidating or scary, if you're responsible for producing quality software?

[00:37:51] LF: I would not say it's scary. For team members in quality engineering or software developers, you are constantly learning something new, and you constantly adapt. And you do not think of, “Oh, I just acquire this knowledge of this tool, and I'm done for the rest of my life. I don't need to learn anything else. The whole system is going to work perfectly.” Absolutely not. In technology and software development and software testing, we're continuously learning something new every day, and we're looking for some new tools that will help us to improve our processes. With continuous integration, delivery deployment, we're still looking for a way how can it be better? Is there some better tool? And we're on the lookout for the most advanced tool or the technology that simplify our process, because the goal is simplicity.

[00:39:00] KP: Makes sense. Is there anything you think we should have gotten to that I haven't asked you about?

[00:39:06] LF: I just want to mention, again, about our company that we're looking for lots of engineering roles, and we welcome everybody. We are very diverse organization. We have people of different races, different background, different ages, different gender, and we welcome people from LGBTQ community, from artistic associations. We practice servant leadership, collaboration, and we're open to people with their own unique ideas, no matter at what position you have at the company. We always welcome people to bring something new. It's fun. We have all sorts of fun activities, and it's just very supportive organization and we welcome everybody.

[00:40:10] KP: Well, those are inspiring things to hear. Liliya, to wind up, I want to ask you, in all these topics, as we've mentioned, a couple times, things have changed really fast. Technology is moving quick. You got to keep your eyes out for new tools and all that. You've done a great job of that. Can you give any advice to listeners on how they can keep up with the quickly evolving field?

[00:40:33] LF: I would say listen to Software Engineering Daily.

[00:40:39] KP: Thank you.

[00:40:42] LF: And attend different webinars, attend different meetups when we'll go back to normal. On LinkedIn, I would recommend people to follow me on LinkedIn and connect to me. If you have any questions you can reach out to me. You can just search for Liliya Frye. You'll find me there. Also, read the on LinkedIn any other posts people talk about, latest technology, different strategy for software development or automation testing. There is a continuous process of learning. So, I advise our listeners to be open to new ideas. to new ways of doing things and be fearless to try new things.

[00:41:34] KP: Great advice. Well, Liliya, thanks again for coming on Software Engineering Daily.

[00:41:39] LF: Thank you.

[END]