EPISODE 959

[INTRODUCTION]

[00:00:00] JM: Web applications are used on a wide variety of platforms. On each of these platforms, the web app news to load properly and allow the user to navigate the website and interact with all of the user flows, such as signup, login and the various read and write operations that make up the functionality of any website.

It's difficult to ensure web application functionality across all platforms because there are so many different configurations of platforms. You've got different operating systems, different underlying hardware, different browsers, different device form factors. These all create potential sources of sub-optimal website functionality and performance.

Testing web applications often involves the work of a manual quality assurance employee, or QA. The QA can simulate the procedures that a normal user would go through. This QA process ensures that the website is operating as expected, but the manual workflow can slowdown software development.

Gabriel-James Safar is a software engineer and the founder of Madumbo, which was acquired by Datadog. Madumbo was founded with the goal of making web application testing simpler by identifying errors in pages and enabling users to create test suites from recordings of user activity. The process simplifies and accelerates the testing process, and those recordings can just be made by a QA person who can make the recording and then rerun the recording overtime rather than having to automate tests, or have programmed tests, or repeat the process manually.

Gabriel-James joins the show to talk about his experience building Madumbo and his perspective on the modern application testing process. Full disclosure; Datadog is a sponsor of Software Engineering Daily and Datadog now owns Madumbo and is integrating that product into their own product.

[SPONSOR MESSAGE]

[00:02:15] JM: Monday.com is a team management platform that brings all of your work, external tools and communications into one place making cross-team collaboration easy. You can try Monday.com and get a 14-day trial by going to Monday.com/sedaily. If you decide to become a customer, you will get 10% off by coupon code SEDAILY.

What I love most about Monday.com is how fast it is. Many project management tools are hard to use because they take so long to respond, and when you're engaging with project management and communication software, you need it to be fast. You need it to be responsive and you need the UI to be intuitive.

Monday.com has a modern interface that's beautiful to look at. There are lots of ways to use Monday, but it doesn't feel overly opinionated. It's flexible. It can adapt to whatever application you need, dashboards, communication, Kanban boards, issue tracking.

If you're ready to change the way that you work online, give Monday.com a try by going to Monday.com/sedaily an get a free 14-day trial, and you will also get 10% off if you use the discount code SEDAILY.

Monday.com received a Webby award for productivity app of the year, and that's because many teams have used Monday.com to become productive. Companies like WeWork, and Philips and Wix.com. Try out Monday.com today by going to Monday.com/sedaily.

Thank you to Monday.com for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:04:07] JM: Gabriel-James Safar, welcome to Software Engineering Daily.

[00:04:08] GJS: Thank you. Nice to be here.

[00:04:11] JM: Yeah, it's great to have you. The first job that I ever had as a software engineering intern was writing scripts, and these were scripts that would test a web application.

So these scripts would run through common scenarios that an application might go through, like a sign-in flow or perhaps navigating a webpage to make sure that I could click into certain things and these automated tests would run on a regular basis. This was maybe 10 years ago. Do modern software companies have QA positions like this even today?

[00:04:52] GJS: Yeah, they do. If you want to know, they even have QA positions with people that just do it manually, and that's click on the buttons themselves. Not even automating it. The answer is yes, your issue is still very, very problem of today.

[00:05:11] JM: Tell me how the modern QA department at a software company works.

[00:05:16] GJS: Actually, the dev word changed drastically with the DevOps movement. Obviously, now operation and development are intertwined and everyone knows it. For the QA, it's not that obvious, and I think that still today many companies are struggling with how to handle the QA, and that's true for basically all the companies, I think, but the bigger you are, the most difficult it is.

When I'm saying that, it means that there are tons of ways that it's handled. Some companies say that they shouldn't have anyone in charge of the QA, because engineers are in charge of doing so, and with CICDs, it's easier and easier all the time. Others are seeing that it's the job of both engineers and product managers to work on QA, and then a very large number of companies are still relying on people dedicated to that job.

I don't want to say non-accurate numbers, but people with ISDQV, the quality software testing certificates several hundreds of thousands. We are speaking of many people that are doing that as their jobs. There a lot of ways to practice the QA even today. In many cases, it's very difficult balance to find, because it can really slow down the pace on which the deployment that the new features can be shipped.

[00:06:44] JM: How does the information propagate through the organization relative to that testing? For example, how does the QA person know what to test? How do the bugs that they discover in the QA tests propagate back to the programmers who are writing code that fixes things according to those tests? Tell me how the workflow goes.

[00:07:09] GJS: Yeah. I will describe you the workflow of how it's used to be, when 10 years ago you were doing that. The Agile way of managing a delivery was not quite there yet, and back in the days, as you know, there were waterfalls. Some people were doing the specs. They were giving it to engineers and then engineers started coding.

At the same time, there were QA people that started doing testing plan. They were listing all the features that needed to be tested. Then whenever the first version of the product was ready, the engineers would put that on the staging of pre-prod environment and then they would have the QA people that would swarm on that environment and do all the tests usually manually, and they were using very complex products that are still very up-to-date, like ALMs, like HP Enterprise products, like now Micro Focus. They would be using that to put every single item, every single test that would succeed. Then it would go back to engineering that would fix that. Then deploy again and over and over again. Usually, after that, they were another way.

So people started to have several waterfalls intertwined, 3, 4 waterfalls intertwined in order to not waste times while the QA was done over the new feature that are released, etc., etc. There was that way of working that's fairly complex with a lot of tooling to work on that. Then all of a sudden, with Agile, something broke up, broke up because companies such as – In modern companies, like we do for example in Datadog, engineers will potentially deploy every day. Each of them might deploy every day. Now, how can you work with QA that's sufficiently agile? You do not even have time to do that least of features to be tested. All of a sudden, it's super hard to do.

[00:09:19] JM: These people who are writing QA tests today, do they need to be programmers? Because back when I was doing this script writing, you definitely needed to be a programmer to have high leverage. I mean, my work relative to the manual person, I mean, I couldn't test things as complex as the manual QA person, but I could write tests that were reproducing themselves so that they could be executed with the click of a button. So it's higher leverage.

But people who write QA tests today 10 years later, we have better tooling. Do we still need programmers writing QA tests?

[00:10:09] GJS: The issue now is that we have a shortage of people able to programmer and

willing to do QA. In some organizations, yes, you definitely do, and these guys are really, really important in these organizations. In many big organizations, they will be – Is a very important all contractors, because it's potentially very hard to find the skills, because doing test automation, as you said, implies that you are able to code. At the same time, it might be seen as not as exciting to code test automation as to code a new feature. How to attract someone with that kind of capabilities?

At the same time, as that kind of capabilities were needed, there were some new toolings, some new tooling that came out, and this tooling essentially aims at lowering the bar of the skills needed to do test automation. It's a global trend, right? It's what we were trained to do at Madumbo and that we are still doing at Datadog. But it's also globally all the movement of RPA, robotic process automation, with great products like UiPath, for example. It's fairly different, but they are not addressing the same use cases. But still, the goal is to make it easier and to have less technical people and potentially more business-oriented people automate stuff, automate tasks.

In the case of QA, yes, the goal is to lower that. Is it lowered? It really depends on the company. There is no one answer, one-size-fits-all answer, but I think that the tooling is now more and more efficient to all the people that do not code, including product managers or QA practitioners to do test automation without having to spend time on coding. Yes.

[00:12:04] JM: You mentioned an acronym, RPA. Can you define that?

[00:12:09] GJS: Robotic process automation. It's trend, or actually it's a set of software that aim at automating tasks that's used to be done by a human being and that are time consuming. There are some great products that are on the market and specialize in that area. Maybe the most famous one is UiPath. I guess you've heard of it. Have you?

[00:12:31] JM: I've heard of it. I don't know what it does. I haven't looked at it.

[00:12:33] GJS: You are not the target. The target are mostly business people who trying to make the process more efficient. It can be very low-tech. It can be like how do I automate checking the compliance of a dog and saying, "Okay, it's okay." That can be that kind of thing.

The goal is not to have to change all the interface of your system to be able to automate it, because engineers could automate everything, but potentially it will imply revamping the system deeply. A way to avoid that is to use these tools to automate that kind of tedious tasks.

[00:13:14] JM: These tests that we're writing in one way or another, when do we use these tests? When do they get run? Do they get run nightly? Do they get run part of continuous delivery pushes? How does QA testing fit into the lifecycle of software?

[00:13:37] GJS: I think that it's now fairly common and we see it in Datadog. It's fairly common to have tests that are running against production, and that's not new. That's something that was done probably 10 years ago when you were doing that. These tests were synthetics. So scripts automating tests made against production in order to detect whenever something stopped working.

Instead of having to monitor all your servers, all your services, etc., etc., you would use the last layer to see if down the road it works. That's very powerful, because whenever a test stops working, whenever there is an alert, then you can be warned and you know very specifically what stopped working, that the feature 'blah' making logging and then creating account and then adding an item to the cart. That path stopped working, and that's very explicit. That's one thing, running against production.

Another very common practice is to run tests after deploys. That's very exciting, because you deploy and then you launch your test. You can do it, for example, against your staging environment. Instead of having a broken code going to production, or instead of having human beings shaking manually that it works, the first layer of job is done by these scripts and tell you if something stopped working after your new deploy.

Another possibility is to do that within your CICD, within your CI pipeline, and that's sort of the holy grail that of course it requires much more tooling, because it means that you have scripts that can access your CI environment. That means that your CI environments is building a real frontend with real URLs you can run scripts access to.

All these things are more tooling, but of course, blocking that code to go to the next environment

is something super powerful and very, very appealing. A vision that I have is that the tooling will make it easier and easier to set up and make sure that kind of feature will be more widespread.

TO give you a secret, until very recently, we didn't do it at Datadog either because creating the setup with security constraints was too tedious. Now that we have the right tooling to do that, we are doing it and we are trying to do it at scale with more and more of our features. Does all that make sense? What do you feel?

[00:16:26] JM: It does. I'd like to talk about the company that you started, which was eventually acquired by Datadog. This company is called Madumbo. What was the goal of Madumbo?

[00:16:39] GJS: The goal of Madumbo was to help people with their QA process. I'm seeing people on purpose because our targets were either the business side in charge of an IT process, like product management or product owners, or the IT part in charge of the delivery. These were our targets. Depending on the companies, the reporting lines were different. But these guys were – We were talking to and we started working on that because we had started another startup with Microfunder, and we had a lot of issues with regressions.

We tried to create the tooling needed to make life easier to everyone, and we started with creating a tool that was, as far as we knew fairly new, that we named [inaudible 00:17:37]. It was some sort of, let's say, a link crawler, but instead of stopping at crawling links, it would trigger every single piece of JavaScript it could find on the website. You can imagine it as something that goes on the first page of a website and then try to find every single state of the website in order to click on every button to take every content to check what's the quality of that content. It will check the spelling, check the responsiveness, check the JS errors, etc., etc., of the product that it started working on totally autonomous manner.

One of the big challenges in doing that was – Technically, it was how to make sure that we had a good coverage. Notably on the forms, filling forms is very difficult. Understanding when a script arrives on the form, that it needs to put the data, that it needs to put a name, city, etc., etc., is fairly tough. Then the other issue was to show the value, because of course it's valuable to show that you deploy, you have a script that covers everything tells you every new error and every new typo that you have. But it doesn't remove a process that you already have.

Very quickly after starting that, we started working on another product that was adding up to the first one, which was a recorder. With that recorder, we tried to say to the teams that they could very easily automate the QA. That for the rest, the other scenarios that they didn't think, they could use the [inaudible 00:19:23] to cover up everything. That was sort of our approach, and very fast it proved to be very, very exciting as a topic, because there are tons of technical issues to tackle when working on that.

[SPONSOR MESSAGE]

[00:19:45] JM: Today's show is sponsored by Datadog, a scale, full-stack monitoring platform. Datadog synthetic API tests help you detect and debug user-facing issues in critical endpoints and application. Build and deploy self-maintaining browser tests to simulate user journeys from global locations. If a test fails, get more context by inspecting a waterfall visualization or pivoting to related sources of data for troubleshooting. Plus, Datadog's browser tests automatically update to reflect changes in your UI so you can spend less time fixing tests and more time building features.

You can proactively monitor user experiences today with a free 14-day trial of Datadog and you will get a free t-shirt. Go to softwareengineeringdaily.com/datadog to get that free t-shirt and try out Datadog's monitoring solutions today.

[INTERVIEW CONTINUED]

[00:20:53] JM: As you said, this was partly inspired by some previous work that you had been doing, a previous company you were working on. You said you were encountering lots of regression, and Madumbo was in some ways an effort of building a company that could solve those regression issues. Can you go back a little further? Can you tell me what was going wrong? What were those issues that you wanted to address with Madumbo at your previous company?

[00:21:23] GJS: Well, we were a very small company and we had basically no resources, but we were obviously cloud native and there were two or three people coding, two or three

developers and these guys were pushing code and putting it to production. Of course, we didn't have anyone to do the QA. We could not waste manpower on checking it manually.

It was a B2C product, and it was basically booking.com for running events, marathons and stuff like that. Whenever the search button for research doesn't work, you are in a very bad position. It happened more than once. It happened several times. After that issue popped several times, we started working on doing test automation. We used Selenium. We implemented these tests in our CICD, etc. For us, we were very small. It was a big investment. It was potentially several sprints of engineers to do that.

It was a big investment. Because setting up the environment takes a lot of time. Setting up the infrastructure to be able to run the test takes a lot of time. Maintaining the test takes an awful amount of time, because one of the big issues test automated by engineers is that whenever the color of the button changes, potentially it breaks a test. So a lot issue happens.

That was the situation we were on, and I think we are not the only ones in that situation of I have a product, I want to go fast. But very fast, I understand that trying to go faster will just break everything and put me in a very situation.

[00:23:14] JM: Explain in a little more detail how the products that you built at Madumbo improved the QA process.

[00:23:24] GJS: Sure. I'll start with the recorder. Essentially, the way test automation works is that you have a script that will be on a Chrome or whatever browser, and that script will try to find the elements it needs to interact with in order to a click or to validate some sessions. For example, most of the actions can be summed up as I want to click on that button and then I want to make sure that it's written "Welcome Mr. Blah. You arrived on the website."

The issue is how to identify the elements. That's a very tedious one. You can use the IDs, but in HTML, there is no validation made by the browser. The browser will always render something even though there are 20 elements in your webpage that have the same ID, it will work. You can do very, very terrible stuff in the frontend. It will work.

The issue is how to identify an element uniquely to make sure that that's the right one to interact with. Another issue is that frontend code is very easy to change. It's much easier to iterate fast. Frontend engineers can usually work on 15 tasks in a sprint during one week. There are a lot of times where you release this new code and potentially can break stuff and it can also change the XPath of that element, which is another way to identify the elements to interact with.

Long story short, it's how identify element, and it's super easy to change their name so to speak. We tried to address that first by making a recorder that is super easy to use putting it in the cloud. All that made it super easy to set up, super easy to create your first test. You didn't have to spend a lot of time setting up the infrastructure to do that, because it's tedious.

Then our main piece of work which was about identifying elements. So actually we created an algorithm able to identify elements with several different algorithms. These algorithms are named locators and each of them are identifying the elements you were clicking on in a unique manner. Then we replay your tests, these locators will try to find the element on your page. Potentially, your button will have changed its color. One locator will potentially rely on the color, because it will rely on classes or on attributes. Then you change the color, but that's okay, because we have other locators not using the same elements or trimming them in a different manner that will be able to identify that element.

Then all these keys are fourteen and seeing which element is the one that they feel is the one they should interact with. After doing all that, they decide on which element the button should click. Then at the end of the test, if the test works until the end and one of the key has broken, then we will re-heal the broken key, leveraging the ones that works. It means that your tests are less brittle. They are more adaptive to what's happening to your frontend and they are also more – Easier to create and faster to re-heal and easier to – They won't break every time.

[00:27:10] JM: One product that you built is this recording tool, and to me this product, if you compare it to the work that I did as a QA person and the manual QA person, it fills some gap between those two roles. For example, I had to write a lot of code in order to have reproducible tests that could run every night or whenever there was a continuous delivery push.

The manual QA person would go through a manual QA flow and report any bugs associated

with it. But their work was not reproducible. The recording system that you wrote is a system where if I turn on the recording and I'm one of these manual QA people, I can then go through the flow and the recording tool turns my manual work into a programmatic test. Am I describing how that tool works correctly?

[00:28:24] GJS: Absolutely. Actually, what you're describing was absolutely what we were preaching. What we were saying is that by making it super easy to use, we load even manual testers to take that in charge, which is fortunate, because in most cases, the people describing the scenarios to be tested are not in the IT department. They are product managers, product owners or reporting to these guys. Usually, it was a way to empower the people and making sure that they could – As fast as they could describe what was supposed to be tested with themselves.

Save a lot of time doing it and consequently increase the quality, because of course you cannot test everything. You can test every single element with an automated test, because sometimes you need just to explore your product in order to feel something is off. The goal is how do we save time to people who are doing test automation to ensure that they can do more testing, that they can do exploratory testing. That's exactly the story.

But I'd like to add something on the fact that recorders already existed. We were not the first ones doing a recorder, but recorders always had that issue that the ways to identify elements that were products by the other recorders was super brittle. Making the engineers very skeptical about using them, because an engineer would tell you, "Look, if I need to use a black box tool and then it doesn't work, then I cannot do anything with it, and then I need to change it and to id manually. So what's the point?" That was why we focused on put so much energy on making that multi-locator system.

[00:30:21] JM: So the workflow once I have a recording system, does that totally enable the manual QA person to have the same degree of empowerment as the programmer who is writing automated tests, or are there still kinds of tests that a recorded manual QA person cannot accomplish?

[00:30:48] GJS: There are several things. In order to ensure to empower them, we did a lot of

efforts. One of the big issues that you have when you are not coding yourself is that you cannot reuse the same scenarios, and if you have a long journey that you want to test, then it's a little tedious, right? Even if it's recorded, even if it's clicking on 40 buttons, still it's a little tedious.

If we wanted to improve that, we needed to introduce new features to do that. We introduced a feature named Subtests, meaning you could use a test within another test. Then we understood that there were issues about how can I check at the same time one tab and another tab? That's very important if you have an admin dashboard and you are performing actions and you want to ensure that these actions are updating the admin dashboard in the proper manner.

We had to add a feature making that validation. Then we ended up topics where people wanted to upload files. They had to add a feature to upload files, etc., etc. What I'm saying is it's a race to ensure that we able to provide all these features. It's a race that's not over, but our goal, yes, is to make it super easy.

At the very end, there will be some things that will still be easier to test with people that are especially standard can code somehow. For example, if you have a system that is giving you points, issue purchase $100, then you get 100 points, the type of systems. If you have that type of system and there is an operation to transform that $100 into two times less points and run them down, then you need a system to write it down in the script. So you need to introduce some JavaScript to do actions against these variables. These things still need people able to code a bit, not necessarily that much, but at least a little bit to be able to automate it.

[SPONSOR MESSAGE]

[00:33:10] JM: When you listen to Spotify, or read the New York Times, or order lunch on Grubhub, you get a pretty fantastic online experience. But that's not an easy thing to pull off, because behind-the-scenes, these businesses have to handle millions of visitors. They have to update their inventory or the latest news in an instant and ward off the many scary security threats of the internet.

How do they do it? They use Fastly. Fastly is an edge cloud platform that powers today's best brands so that their websites and apps are faster, safer and way more scalable. Whether you

need to stream live events, handle Black Friday traffic, or simply provide a safe, reliable experience, Fastly can help.

Take it for a spin and try it for free for visiting fastly.com/sedaily. Everybody needs a cloud platform to help you scale your company. Everybody needs a CDN. Check it out by visiting fastly.com/sedaily.

[INTERVIEW CONTINUED]

[00:34:24] JM: So if I'm an automated or a manual tester that is using the recording the tool, let's say I'm going to test a form, like a form to fill out a mortgage or something. I click the recording tool and I start going through filling out the application, and then I hit submit, and it sends me an email with the application filled out. That's the end of my test.

I've got an end-to-end test now. It's completely recorded. What happened beneath the surface in that test run? What does the browser do when you click the record button? How is it recording the behavior of the person who is clicking around in that browser window?

[00:35:16] GJS: The way we decided to do it was to display your website on which you want to record within an iframe. It's a choice that we made, because it allowed us to give you very good visibility over the steps that you are creating as you are creating them. As you are doing that, we are creating a JSON of instructions that are the steps and that are describing what needs to be done by the script. It's very simple actually in terms of – It's a very simple JSON describing the steps.

Then once you run it, we use, first of all, the configurations asking what's the stat URL, what are the locations from where you want to run it. What's the size of the screen, etc., etc. Then we run it on a browser using Puppeteer to be able to automate the testing. Puppeteer is a way that makes it much easier to create tests. For us, it means that it's easier to create these steps and to run them and we have a driver executing these steps.

[00:36:19] JM: So your company, Madumbo, was eventually acquired by Datadog. Why did

Datadog acquire Madumbo. What are the synergies between a logging company and a company that does automated web application testing?

[00:36:36] GJS: As you probably remember, Datadog started with monitoring infrastructure. Datadog started with, “Okay. The cloud is coming and we need to be able to monitor these new types of hosts that are arriving.” Datadog strived on that market.

Then after a few years of being successful on the infrastructure monitoring, Datadog started working on application performance monitoring. It entered another business domain that was already fairly crowded with great companies. Then we decided to be differentiated compared to them by adding the logging layer. I think you spoke already with the VP of product of a company named Logmatic that got acquired by Datadog, and that was working on that logging system.

Datadog had three pillars of observability allowing people to monitor very, very thoroughly their backend and infrastructure and to manage their logs, etc., etc. But first of all, our Datadog’s APM competitors had some solutions to monitor the frontend. They were allowing people to do synthetics to run tests against production to ensure that it worked.

These competitors had the product at an offering that Datadog didn’t have. At the same time, Datadog has always been trying to break down siloes between different personals. First, it was between developers and operational practitioners, and now with us, with Madumbo, with the acquisition of Madumbo, Datadog is trying to bring new personnel that are also working very closely with the IT team, which are the two-way practitioners and product managers. All these people who care about the product as it’s seen by their customers, and the goal is to bring all these people on the same platform, because we believe that it makes the process of QA and of ensuring that the product is in a good situation much easier when you have everyone within the same platform.

Our goal essentially is now, and I think that was the strategy when we joined, was to ensure that you could have people creating test scenarios and then you could have engineers using these same scenarios in the context of monitoring operations, monitoring production, or in the context of deployment, etc., etc. How can we free up time to engineers so that they can focus on their features and at the same time ensure product management that the quality is always at a

certain level? That's the value brought by having all these solutions on the same platform.

[00:39:32] JM: What have you done since the acquisition?

[00:39:35] GJS: Well, in Datadog, as I told you, our goal is to have one platform. It means that we had to redo everything we had done within Datadog, and that's a lot of work, because it means that we have to create – Well, to recreate everything, but to create it within a very big and well-oiled machine and we also had to integrate with older features, which was a lot of work. We also decided to create another product that appeals to the infrastructure practitioners and SRE people that are very much the core audience, or used to be the core audience of Datadog by working on the testing of APIs. We are not focusing solely on monitoring the web journeys, but also on monitoring APIs, because for many companies APIs are also their products. Their products are not only the web UI, but it can be also the APIs underneath.

So with all that offering, we had to work in order to make it integrated. We worked very hard in integrating it with the backend and the infrastructure in order to give you whenever you are launching your tests to give you the state of the backend, the state of the infrastructure at the same time, so that if a test stops working, you can very easily understand if the issue comes from the frontend with screenshots or with JS errors or with resources that we collect from the backend, because we show you the backend traces associated with it or with the infrastructure, because with the backend traces, we can go down to the host that run that very specific piece of backend code.

That was a lot of work to ensure that all these elements from the very end-user experience to the host executing the backend code, everything is tied together and everything is working fine together. That's been quite challenging and very exciting. We also discovered a scale that was totally different from the one we had at Madumbo, because we were focusing on fairly big enterprise companies, and with Datadog, we could very fast work with hundreds or thousands of companies that are willing to use the product, which means that we have to have infrastructure on our side able to run all these tests. That's fairly challenging to do all that.

[00:42:13] JM: As an acquire, what are the keys to making an acquisition by a bigger company successful?

[00:42:20] GJS: I think that first, at least that was our case. You need to look up to the people who are going to be your new bosses, right? We were really impressed by the executives of Datadog that we met. We were like, "Whoa! These guys have a vision. They know what they are doing and they have that platform vision and they see why they want us to be involved. I think that might be obvious, but I think that it needs to be checked."

Then I guess that what made us very happy with joining Datadog, was the freedom. How is the company organized? Are they willing to give you freedom and to let you work on the project that you are willing to bring by guiding you in order to make it in line with the rest of the product but without telling you exactly what to do? Are they going to just explode your team, put it within various teams, destroy your product because for some reason there were some change in the alignment of what's being done, things that needs to be checked. Some friends of mine that got acquired didn't end up that happy with the acquisition, because potentially they couldn't necessarily keep doing what they were really thinking was valuable for our customers. That's something that we were fairly happy with after joining Datadog. That was the two things.

Then maybe the organization, Datadog is a very agile company. Even if we came from a super small startup, we were not lost. We were not with processes that would make everything hour long to decide where to put a button or – Yeah, everything was smooth, essentially. That's another thing that made all our engineers very happy. As you can guess, people are key in engineering. We need engineers to be involved if we wanted to be able to create products. Our team stayed fairly onboard. They are still all here more than one year after the acquisition. We got fairly successful on that. I think these three points are the main thing. Look up to the top management, making sure that you will have enough freedom, and then making sure that in terms of working habits, everything is sufficiently aligned.

[00:44:41] JM: In your time building Madumbo, what were the biggest lessons you learned from starting a company and taking it to an acquisition?

[00:44:51] GJS: That's a tough question. You learn tons of stuff. Well, whenever you start doing something of your own and you haven't done it before, you learned to fight to get what you want. That's probably the main thing. If you want to sell something, you need to make sure that

it happens. So you need to fight to make it happen. If you want to make a great product, you need to fight to make it happen by asking people for feedback constantly and asking them what they think is really the key issue for them.

We were lucky enough to be working with educated people that are understanding what we are trying to do. Asking them what they wanted and how they wanted us to do it has been probably the biggest lesson. Listen to your customers is something that is absolutely key, especially in the software industry and in the industry of monitoring software. That was probably amongst the biggest lessons.

After the acquisition, we also discovered the scale all of a sudden, and that's something that we are fairly impressed by, because going from one size to a size which is much, much larger, so fast, it teaches you a lot of stuff about how to scale up support, how to scale up outages processes. How to scale up the engineering team, etc. All these things we learned very fast by doing so.

[00:46:19] JM: If you ever start another company, what will your next company be?

[00:46:23] GJS: Oh! That's a super tough question. There are tons of great things to be built. To be honest, at the moment I'm fairly focused on building Datadog's [inaudible 00:46:35] and RAM solution, because as you might have heard, we are working on real user monitoring. These topics are super exciting. At the moment, I'm working very, very hard in making it happen. But globally, I really think that there are a lot of work to be done around how to automate processes, because a great share of the world that we do constantly is very repetitive. So probably, there is a lot of room for improvement in that area. But once again, I didn't think about that thoroughly enough and we are fairly focused and happy with working at Datadog.

[00:47:14] JM: Gabriel-James, thank you for coming on the show. It's been great talking to you.

[00:47:17] GJS: Yeah. Thanks. That was great talking to you as well.

[END OF INTERVIEW]

[00:47:29] JM: DigitalOcean is a simple, developer friendly cloud platform. DigitalOcean is optimized to make managing and scaling applications easy with an intuitive API, multiple storage options, integrated firewalls, load balancers and more. With predictable pricing and flexible configurations and world-class customer support, you'll get access to all the infrastructure services you need to grow, and DigitalOcean is simple. If you don't need the complexity of the complex cloud providers, try out DigitalOcean with their simple interface and their great customer support, plus they've got 2,000+ tutorials to help you stay up-to-date with the latest open source software and languages and frameworks. You can get started on DigitalOcean for free at do.co/sedaily.

One thing that makes DigitalOcean special is they're really interested in long-term developer productivity, and I remember one particular example of this when I found a tutorial in DigitalOcean about how to get started on a different cloud provider. I thought that really stood for a sense of confidence, and an attention to just getting developers off the ground faster, and they've continued to do that with DigitalOcean today. All their services are easy to use and have simple interfaces.

Try it out at do.co/sedaily. That's the D-O.C-O/sedaily. You will get started for free with some free credits. Thanks to DigitalOcean for being a sponsor of Software Engineering Daily.

[END]