Can DevSecOps Make Developers Happier?

featuring Hasan Yasar as Interviewed by Suzanne Miller

Welcome to the SEI Podcast Series, a production of the Carnegie Mellon University Software Engineering Institute. The SEI is a federally funded research and development center sponsored by the U.S. Department of Defense. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

Suzanne Miller: Hello, my name is Suzanne Miller. I am a principal researcher here in the Continuous Deployment of Capability Directorate, with my colleague and my boss, Hasan Yasar. He has been a frequent guest of ours on the podcast series, and I want to welcome him back. So, welcome Hasan.

Today, we are going to talk about, can DevSecOps make developers happier? This is a little different conversation than we often have, but this points to the cultural aspects of adopting DevSecOps, and so, I wanted to give Hasan a chance to talk to us about that today. Before we get started with that, Hasan, please give us just a brief, for those who have not met you before, introduction as to what you do here at the SEI and what brought you to this kind of work and doing this kind of work at the SEI?

Hasan Yasar: Suzanne, thank you so much for having me. It is another pleasure to talk with you. I always enjoy talking with you for any condition, thank you. I am a technical director at SEI, but I have been working at SEI almost more than 10 years actually, time is flying, 10 years. And why I came to SEI, I was working in industry for 20 years. I did a lot of engineering, support engineering work and writing code for various environments, various technologies. I ended up really using my knowledge and experience in research and also the academic domain, like really helping the world [understand] how can we connect between researchers and practitioners? There is always disconnect in between. I ended up myself in SEI and exposing and getting my learning pieces into the communities. Then after starting at SEI, I realized that we are not really following so much modern software engineering practices. That is how industry is going in terms of software deployment, development, all these capabilities. When I started the first time in 2010, I built up the first DevOps pipeline in 2010.

Then we started to really push the boundaries in how SEI can help mainly DoD and also the rest of the industrial practitioners to build up engineering practices, using modern tools and techniques by thinking engineering thoughts behind it. It is not just the tools, talking about what the right engineering concept is. We can use the right tool, right methodologies.

That is why I ended up in SEI, and I also really enjoy working with the communities and working with our sponsors, as well as the people that need help to identify what needs to be done for engineering practices. Mainly, we are talking about DevOps and DevSecOps. Maybe next time we are going to talk something else. We may say AIOps [Artificial Intelligence for IT Operations], who knows, we may say something else, but engineering practices [are] not changing, they are evolving with better automation and with speed relevancy and also improving [to become] more business oriented.

Suzanne: All right, so when we are talking about those practices, we are talking about DevOps as a multi-dimensional way of understanding how to go through a pipeline from concept to capability, which is how we basically talk about it. I wanted to use Dan Pink's model of talking about happiness, but from the viewpoint of, what are the things that tend to make knowledge workers in general happy in their work?

The model has three components. One is autonomy, one is skill mastery, and the third is purpose. So, I thought that would be a nice way of framing, how does DevSecOps improve the developer's experience in all three of those areas? Let us start with autonomy. How is it that DevSecOps improves a developer's ability to be more autonomous, to have more control over their decision-making environment and the development environment?

Hasan: It is a great way to start actually, so I am going to give a little bit of background for DevOps and DevSecOps. It is a lot about automation. There are four principles we have been told many times. The four principles on DevOps are infrastructure-as-code pieces, automation specifically, and culture and communications, and monitoring. To achieve all elements, automation is the fundamental component in the DevOps perspective. Like, what is fundamental? If you are talking about the feedback mechanisms, they have to automate the way to share the outcomes from the CI [continuous integration] server or outcomes from any static-analysis result that can be shared.

Automation is playing a key role for sharing. Automation is playing a key role for [doing] manual steps [in] automated fashion, like test cases or test scenarios or running multiple security testing or running multiple functional testing as well. The best part is I really like [those] pieces [because I know how painful it was doing these things manually for many years]. It is a deployment perspective. I remember when I deployed an application creating MSI files years ago, a decade ago or more than that, creating MSI files, which are installers, then packaging up,

and then trying different environments, different versions of Windows. I remember I set up five different machines with a different Windows deployment to try out if my MSI is going to work on that. It was a pain. It was really a pain. Now with automation, it is really helping me try out different environments, different platforms, and saving my time. When we look at that perspective, it is helping my time to automate that process.

Second, I can see the result immediately if it is going to work or not, what I have to do because a troubleshoot[ing] perspective is also important. If you have a manual step that you do for a deployment for testing, it is very challenging to go in depth and find out what is not working, what type of problem you have versus if you have done [it] automatically, then the system or the server or the automation script will tell me if it is going to work or not, and here are the problems. Makes me really happy because I can see the result immediately. There are things we can discuss, but that is why the automation perspective saves my time, taking the manual steps and saving my time from an automation perspective.

What is the outcome of that automation? Makes me think about a new concept or learning fast or getting a result, getting an impact, so we can expand that effectiveness. It is like I am not doing the same thing, which is manual, over and over again. I do more automation.

Suzanne: Another concept that we talked about, that comes from the work of Matthew Skelton and Manual Pais on team topologies is cognitive load. What you are describing is you are removing some of the extraneous cognitive load that things like having to have five machines to deploy your install environment is going to create. It is not really focused on the application. It is just focused on the extraneous aspects that have to be dealt with, but if you can automate those, you do not. The other piece of autonomy that I wanted to look at a little bit is governance. Because one of the things that I have observed in DevSecOps environments is that if the governance structure, if the management structure of the organization trusts the automation, and trusts the kinds of decision-making criteria that have been put in place for successfully moving something forward, that is another aspect of autonomy. Because if I do not have to go to a change-control board after every step and have a human look at this, I really do gain back some autonomy as I pass my work through the system. If I fail fast, I fix it, and I move on. But as it progresses through those criteria and makes it through the pipeline, I do not have to do as much interaction with the governance structure to be able to get to the capability deployment. Have you observed that as well?

Hasan: Yes, yes. I think the biggest challenge is for the security perspective that is related to the compliance or the governance role, and how can we pass the guardrails, how can we pass some static-analysis result? Or how can you pass the test coverage risk-factor security? All these we were running in various test tools with the security perspective, like static analysis, as I said, or using a basic application of pen testing. These really take time in getting results. Then based on

the result, which is about a 50-page report, then the person has to negotiate with the security folks to address those items. Negotiations become more political sometimes, more confrontational. If you use automation, that system is going to tell you what needs to be done as a specific vulnerability. Basically, we are taking out the human elements of criticizing or sharing a 50-page report. Now we have very specific feedback that comes from many specific test results from the CI server or any type of test harness. That really helps a team of developers to work on a specific subject.

There is also another side of automation, which goes back to the trust relationship and also opening up more cultural awareness. If my security-team members are sharing artifacts in a digestible way—maybe feedback, maybe a small size of information that the dev team can work as opposite—then that team can share all the infrastructure pieces as scripted. Because sometimes the bill of materials, what my build script has in it, all my dependencies and all the software build components, will be shared as a script.

Script is actually another automation concept because you have something written, which is a script. Sometimes a tool will do the work for us. Sometimes, as a developer, we have to write a script. That script can be a batch script, that script can be just a field configuration item. That [script] can be any type of tools, either the tool input, like maybe a chef, maybe a cookbook. Any tool has some input file. Again, this is a script. The developer can write that script and share it with the security teams, *Here are my dependencies, here is my list of items that I am going to work on*. Basically, sharing those artifacts in automated fashion creates transparency between the two teams. Transparency actually results in trust. I can trust, as a security person, the developer working on it, with a fact. It is not, *Hey, I am working on this one. I am pulling the slider from here and there.* Variability is not accepted in today's technologies because sometimes…from a security perspective, I cannot trust peoples' words, I have to trust what the facts are, because at the end of the day, as a security person, I am responsible and liable to protect my systems.

Suzanne: Sure.

Hasan: Artifacts generated in an automated fashion create trust in both parties. I think we have to look at what will make them happy, and trust is one of the elements. If we trust each other, we are not arguing with each other anymore, we are not in confrontation anymore with people. We are talking based on facts. We are not really making up something. The fact is becoming a key driver and sharing the artifacts on both sides, same for security…

Suzanne: It is a fact-based and data-based decision making…

Hasan: Exactly.

Suzanne: One of the things that enhances trust, and trust enhances our autonomy. Let us move and talk to mastery, skill mastery. How does DevSecOps improve a developer's ability to gain new skills? I have heard you speak of some things right here. What if I have never done the kind of configuration mapping that you are talking about into scripts and things like that? That could be a new skill for me. But what are some other things that you have observed that relate to improving skill mastery, as opposed to when you are talking about people using DevSecOps?

Hasan: Before diving into skill mastery, I would like to say one more thing that goes back to the trust and also the happiness. There was a DevSecOps community survey this year. This year the survey was more about peoples' and developers' happiness. There was an interesting finding this year. We found that if the team is happy with what they are doing, they are relying more on fact, versus teams that are not following DevSecOps practices, they are relying on rumors. Rumors are, he said or she said what the system is. It was really interesting to see that survey, and people are using fact. Facts are used by the happy team members versus grumpy developers, who are using whispers, rumors basically. They are making up something because they do not know.

Suzanne: Using hearsay.

Hasan: Right. Either they are making up something or they are assuming something. In the security world, assumption is the big enemy for us. We cannot assume something. We have to really work based on fact, and the happy developers are working based on fact and are more effective than grumpy developers. It was a survey that we ran this year.

Going back to your questions, the increase in skill is important. Let us start with the automation perspective, Suzanne. If I did more manual steps, if I spent most of my time, for example, setting up five different environments just to test if my script is going to work or not, it does not give me enough time to learn new skills. I am spending the same time, maybe more, on the same work over and over again. If I spend less time on manual steps, it is going to open up room in my calendar, it is going to open up my day. I can look for new concepts. I can look at the new technologies. I can experiment with a new concept, maybe new techniques, and new algorithms. I can get my systems better. If I do more [manual] work, I will always be behind the work, following rework. Versus if I spend more time on new things, I will learn that stuff. If I am going to learn new technology, it is perfectly fine because I have enough time. What we see in DevOps organizations is that the biggest return of investment is in basically eliminating rework. Rework means repeating the same work over and over again or doing the same installation over and over again. Based on the recent survey from Dora Research Institute, what it says is that in high-performance organizations, DevOps organizations, sometimes, they are basically eliminating 90% of rework. It is a huge number. What it is really telling us, if I take the rework from a team at 90%, what will the team do? They are not going to stay doing nothing, they will learn something. They will investigate. They will [devote] more time to learning as well. Another

reference to the survey, happy developers are spending 65% of their time learning, and grumpy developers are spending 45% of their time on learning. Basically, you see 20% difference, then the happy developers are learning more self-paced. They are doing this because they are using more automation concepts, so they have enough time to learn at self-pace. If I spend more time, 20% is a lot, 20% technically is like five days in a month. Five days in a month is [dedicated to] learning. If I spent five days in a month just in learning this stuff, I will increase my skills from various perspectives, automation, maybe learning new technology, learning new tools, anything or maybe getting better at coding and better writing algorithms, another way of improving our skills.

Suzanne: To that point, one of the things that I have also heard from developers is that that infrastructure-as-code piece is actually a learning mechanism, in many ways. Because I can look at different ways that different developers have formulated their infrastructure-as-code environments, learn things that way, and a lot of it has to do with, *Oh you automated that, I did not know you could automate that, oh I see what you did*. And then I can incorporate it into mine. There is a lot of that transparency, I think, is what I want to point out. The transparency of the environments is actually a learning tool, as well as a tool for autonomy and trust. Are there some other things out of the survey related to skills that you wanted to bring up before we move on to purpose?

Hasan: Sure. I will give another example, actually from my students, I have been teaching DevOps in CMU [Carnegie Mellon University] the last five years. I always start with the first assignment and write the steps in a basic text format. Next question was, *Can you convert the steps into an automation script?* The students are struggling—graduate students at the first and second one—but they are learning how to write automation scripts, how to be effective in learning the tools and the scripts that they are writing. It is basically helping them to learn the new concept. At the same time, it is teaching them to basically write a good script by verifying what they have done. It is another problem within software communities. We are always thinking, *My code is going to work all the time*. We do not know it is going to work all the time, unless we have the tests done.

Maybe you have done it. I might have done it all the time. You know? Right, with the automation concept, which is IAC (infrastructure-as-code) concept, it is really telling us how much we are accurate in terms of the writing component. In the script, it is really going to tell either yay or nay. It is going to work or not. There is no excuse. But if your instruction is not clearly written, I may make some mistakes. Maybe the producer can make some mistakes. I am a consumer, I can make mistakes. Basically, it is a lot of integration, a lot of, I write a problem we may see as a result of…

Suzanne: I am back to assumptions again...

Hasan: Right, right.

Suzanne: You know, when I am giving you instructions, I am assuming that you are hearing what I am intending. But you may have a different context, you are not necessarily going to hear it. When I tell the machine the instructions, the machine only has one way of interpreting. And if I did not write the instructions correctly, it will not interpret them correctly, it will fail. We have more boundaries for failure, in some ways, when we are doing human-to-human than we are when we are doing human-to-machine, and that is something that improves our skill.

Hasan: Exactly. Right. For the survey, we discovered a couple other items as well with the happiness and unhappiness. Again, happiness means teams or organization are doing DevSecOps, following the nature of DevSecOps, like automation integration, which we covered. Interesting, we saw this this year as well, about friction between the management team and the dev team. Guess what? Literally, the grumpy developers are 80% dissatisfied with their management because their management is disengaged. Only 10% of the developers are dissatisfied, 80% of the grumpy developers are dissatisfied. There are really huge differences between the teams. Happy teams 90% satisfied versus grumpy, 20% satisfied. It is a huge difference.

Suzanne: Yes, it is.

Hasan: It was interesting because one of the elements for DevOps we have not talked about is friction. Friction is when the management team says something, and developers say something else; again, it goes back to the assumptions. Or there is a disconnect with the leadership because leadership is asking for another component, maybe something [for which] it is not transparent what the expectations are for [some] given work or the schedule. There is always friction in every level for the expectations, for the work, for the deliverables, for any type of schedules or the quality, but there are many objectives, many frictions, we can see that. It was really interesting then that happy developers are far more happy with their management chain versus a grumpy developer. It was interesting.

Suzanne: That brings up alignment. Because if we are not aligned, if we do not perceive as a developer that the business understands what it is we are trying to do for them, that is an alignment problem, that leads to that kind of dissatisfaction, and vice versa, if the business does not believe that the developers understand how to serve the needs of the business, we have an alignment problem. In some ways, I am probably searching a little bit, I may be reaching a little bit. But that is actually part of the purpose aspects. If people do not communicate the purpose of what it is we are building and we do not communicate the purpose of the automation, we do not communicate the purpose of the way we have the governance structure set up, if we do not have

a mutual understanding of purpose, that leads almost immediately to those types of alignment issues.

Hasan: That is one of the reasons actually why people hate, or they do not like, the waterfall concept, because it is creating a friction among team members. What the manager says in a written format, maybe the Word doc or something else, it is not clearly setting the expectation of the work and sharing some artifacts. And dev teams interpreting it in different ways and using different tools and different environments creates friction from the communication perspective. Other friction is the schedule perspective, like the developer estimates something, either overestimated or underestimated, [and that] creates [more] friction for the PM (program manager), program management, or the business owners. *What needs to be done? Where are we, what do we do? What are the current schedules? Are we on time?* It makes [things] stressful. In that situation, of course, people are going to be looking for different perspectives because everyone has their own incentives. DevSecOps is bringing a team concept, and including the management team, including leadership, they are under the same team because their goal is, as you said, their goal goes to the business perspective, and it is our goal. *Here is what we would like to achieve.* Let us look back, let us see how we can achieve our goals that we said we can do in two weeks. Based on the goals, we can look at the work, what needs to be done, and everybody takes a piece of the work and then…

Suzanne: I want to highlight that two-week aspect because when you go back to waterfall, we are not talking about getting through requirements to testing in two weeks typically. We are talking, in many complex DoD systems, we are talking years. The disconnect in alignment often happens because of the amount of time it takes from the conception of it on the business part of the requirements to the actual implementation. The guy who wrote the requirements may be long gone by the time I actually get the specification to develop from. But with DevSecOps, we have compressed that time. In the context of Agile, two weeks is typical, or even one week, but the actual cycle of getting a function through the pipeline and verifying that it works as intended and meets the intent and the purpose of the business happens much, much faster, and now I can correct. If we have a missed assumption between the two of us, I have not spent a lot of time working on it, you have not spent a lot of time trying to explain to me all the nuances of it, we can just say, *Oh, no, it is this not that*, and we can move on. That is huge in terms of reducing the friction that you are talking about.

Hasan: Right, and another friction we have not really touched on are the users. If the user sees the result in a short timeframe, we will understand what the user needs to have. Another thing that actually happens is, maybe I forgot to say it, if I see my code is being used [by] users, it makes me happy. Like, I am proud of it because somebody is using my code. Otherwise, I heard so many stories from our…

Suzanne: *I made that!*

Hasan: I made that, right? I made it. I wrote my code. Somebody is using that. I get a lot of interest. Makes us really happy because creating software is an art. Then somebody has to appreciate that. What is that appreciation? It comes from the users. And usage of the code that we produce makes the developer happy. If you see more usage of the work, [that it] has been used by a typical user, [it makes me happy]. I saw many times in my career, my work history, and so many times, if my code is not used, it makes me uncomfortable. I do not want to go back to that project again. Because I do not see impact on working on it. That is kind of wasting my time. I do not want to waste my time. I would like to get really working on a project [for which] code is being used by users. That makes me really happy.

Suzanne: That is the other piece of purpose is if I can see that what I have done has utility for the intended user or even unintended users, then I have that sense of what I am doing is worthwhile, and I am not wasting my time. That is another aspect. What were some other things on the survey that came out about user satisfaction and that purpose element related to happiness?

Hasan: Interesting one, again, job satisfaction. Happy developers are more satisfied with their jobs. Yes, because they are really happy with what they are doing. And job satisfaction is bringing talented people together. And what the survey says is, 90% of people are satisfied with their jobs as a happy developer, and 60% dissatisfied as a grumpy developer. Interestingly, and out of the response, the 90%, they will advise friends to bring into the organization so they can work together, and grumpy developers only 60%. The happy developer is actually bringing a talented team together because they are happy and attracting their friends to work together as a team, and grumpy developers, they do not. Now we are looking for people with a skill set to work together, and DevSecOps has become a magnet to get a skill-set team together and working in harmony.

You have got other things from the HR perspective. We should really think about what we can do, like technical work is affecting our HR strategies. If a team is working well, if a team is producing well, they are happy, they are able to get other team members from outside of work [to join the team]. It is not a money issue. It is about the satisfaction of the work, actually.

Suzanne: Yes, and that is what Dan Pink's research showed that…and Herb Simon and lots of other people, that you can take money off the table as long as there is enough for you to take care of your family and things like that. Beyond a point, it is all these other things that determine satisfaction and happiness in the job. The autonomy, mastery, and purpose are the big things that knowledge workers are looking for.

In terms of how users interact, the user interaction, there are things in the survey about how the perspective of the user changed or the perception of the developers changed towards their users if they are happy, or if they are, as the survey says, the grumpy developers? Any insights on that?

Hasan: Based on the survey, we find that happy developers, [like those in a] DevOps organization, are critical to deploying to the production environment versus grumpy. On average, they are able to deploy more than 50% per week and deploy an application. And other organizations, either like a non-DevOps organization or the grumpy developers, are deploying into the production environment, at 9%, and then usually they are following every other week about 20%. Happiness is also affecting the deployments. We can deploy much quicker and much faster with the DevSecOps because we can see the result and then get back and respond quickly for any incidents. That is another thing we discovered.

Another one we discovered in the survey, again, it takes less time for response and incidents in happy developers versus, again, grumpy developers. Three days is typical average response time for happy developers, versus grumpy about 45 days. [These are] huge differences. If we are responding to any incident in three days, that makes it really much quicker to [respond with] any work in a short time versus 45 days is really, really long, because 45 days is too much. It is dangerous. There are many examples that we saw. Delaying in a response is going to cost us so much time and money and expose data for some other purpose. It was another mind-boggling [thing] when I saw it.

Suzanne: That difference, I can understand it from the viewpoint that if I am deploying multiple times a week then chances are that I am actually going to be able to get to my code faster, where the source of the problem is. If I am not deploying very often, then it is going to take me longer to get back to where the problem is. I would have expected maybe two weeks instead of three days. Forty-five days though, that is a really, really big difference.

When we talk about all of these things together, we are looking at different aspects of the DevSecOps principles that align with creating happier developers. We are talking about having a culture of trust. We are talking about automation that effectively reduces the extraneous cognitive load, and it reduces the amount of time that we have to spend doing rework kinds of activities. We are talking about the monitoring aspect in terms of being able to see the effects of any changes that we make, so that leads to better autonomy. It also leads to us being able to catch things faster, three days versus 45 days. And infrastructure as code is one of those big enablers that helps us to collaborate and to make sure that all of us are on the same page in terms of what it is that we are actually producing. I guess my point is, all of the elements of DevSecOps contribute to happier developers, not just the automation. It goes back to what the SEI is always saying, that you cannot just look at automation in isolation. You have to look at all of these things.

Hasan: That is exactly true, Suzanne. We do not want to look at one element, it is the combination of all. Like, even when you have a great platform in place, there are many examples I saw in my engagements, people are building a great DevSecOps environment. But they cannot use it effectively because there was a process or engineering work that needs to be done to support that environment. Otherwise you just buy a tool from a shelf and put it in the environment, and it is not going to solve the problem. We have to really look to use the tools effectively, with the effectiveness of the DevSecOps environment. That will help us as a team to solve our problem, which is the manual problem that we discussed, improving the communication as a result or improving the trust between team members, being transparent between team members. If you are using tools effectively, you are creating that culture. Again, it is not a one-day or a week job, it is a journey. And other things, we have to make sure that to go from here to our end goal, it takes time, especially if you are starting from a greenfield environment. We have to start from the beginning. It is much easier versus a brownfield environment, where we are starting from a legacy system, and we have to maintain all the code bases. It is another topic we may discuss. It is difficult on an organization that has been established and then puts in a new DevSecOps concept, even before Microsoft.

Microsoft was known as a software guru. It took them a couple of years to build up a DevSecOps [capability] in their organizations. If I am remembering correctly, it looks like they spent three years to build up a DevSecOps concept with the leadership. Their leadership, it starts from a team perspective, and they said, *We are empowering our users to make sure they can do their job well*. Change the perception, and after that, the technologists support that vision and people support it as well. Okay?

It is a different topic we discuss. But going back to your question, Suzanne, it is not one element, it is a combination of all, combination of the principles with the right processes and the right architecture that makes a really good, happy environment. Then we saw an example in the survey, leadership will be really happy because they can see the results immediately, and the program manager will be happy [because] they can see the on-demand testing and schedules, and the security person will be happy [because] they can trust what the developers are working on. And developers can trust the security team members, because the security team will help them to learn and then advise what needs to be done.

Other things I am going to open up, if you have time, are the learning pieces. The learning in the context of security is important because we do not really teach the security concept in school. We are talking about the engineering principles. We are talking about the software engineering, like algorithms and operating systems databases, and all these components. At these schools, teaching security [is important] because systems are living things. It is required that we keep up the knowledge on a continuous basis. I am not expecting every developer to know every

vulnerability. But the developer knows what to do based on their work, based on the technology. When it comes to security, they have to follow up on the new trends in terms of the new attack patterns maybe, new vulnerabilities, and we see a lot of new vulnerabilities keep coming today. Only the teams knowledgeable about security are knowledgeable. Then what the security team can tell the dev team is, *Here are the solutions you can use, here is the feedback for you to learn*. Basically, DevSecOps is really helping the developer to build up their knowledge with the security perspective. It is kind of like a small-sized learning piece, which is specific with each vulnerability. And then they can learn over and over again. That is another good side effect. It is a positive side effect of DevSecOps.

Suzanne: It allows them to learn patterns. I mean, I think that is…

Hasan: Exactly.

Suzanne: Most of the security folk I know, that is when they feel like they have succeeded is when the developers have gone from needing guidance on individual vulnerabilities to being able to see patterns and being able to predict, if I use this pattern in this kind of code, I have seen that kind of vulnerability before. I do not want to do that. I want to use one of these other solutions that prevents that kind of thing from happening. As you said, the vulnerabilities morph, they evolve. But the patterns of thinking that lead to a lot of them are pretty basic. It is more about the threat actors finding different ways of applying the patterns that have worked for them before, and we sometimes give them those kinds of opportunities with different architectural choices and different coding choices. But it is figuring out what those patterns are and then eliminating those in the way that we work. That kind of learning takes practice, and that is one of the things we get from DevSecOps, as you said earlier. We get more time to practice because we are not doing the extraneous kinds of things like putting five development environments up so that we can test our installs.

Hasan: Exactly.

Suzanne: I want to thank you, again, for spending this time with us. I am hoping that some of the leadership that may see this kind of a podcast will understand that what we are trying to get at here is to help motivate people to actually take the steps because it is a journey, and it does cost resources, and it is something different. But getting to happier developers gets a lot of good business results, not just individual job satisfaction, which as an individual we want, but as an organization, what is it about it that makes us want happy developers? It is because those happy developers give us more positive business results. I thank you for spending this time and helping us to talk about these things so people can get some better ideas about how they can go about this.

For our viewers, I want you to know that in the transcripts that we provide with these podcasts, we will have a reference to the survey that Hasan has been referring to throughout the podcast. You will be able to look at that report on your own. Also, any time, just like always, if you have questions about what we have talked about, please do not hesitate to contact us at *info@sei.cmu.edu*. We will have this podcast available on our website, as well as from all the other places that you get podcasts in today's world. I want to thank you once again for viewing today.

Thanks for joining us. This episode is available where you download podcasts, including SoundCloud, Stitcher, TuneIn Radio, Google Podcasts, and Apple Podcasts. It is also available on the SEI website at sei.cmu.edu/podcasts and the SEI's YouTube channel. This copyrighted work is made available through the Software Engineering Institute, a federally funded research and development center sponsored by the U.S. Department of Defense. For more information about the SEI and this work, please visit www.sei.cmu.edu. As always, if you have any questions, please do not hesitate to email us at info@sei.cmu.edu. Thank you.