DevSecOps for AI Engineering

Featuring Hasan Yasar and Jay Palat

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.

Jay Palat: Hi and welcome to the SEI Podcast Series. My name is Jay Palat, and I am a senior engineer in the AI Division as well as the interim technical director for AI for Mission. With me today is Hasan Yasar, who is the director of Continuous Deployment of Capability at the SEI. Today we are going to discuss the engineering of AI systems with DevSecOps—what that means, the challenges and strategies for organizations who want to learn more. Welcome, Hasan.

Hasan Yasar: Jay, thanks for having me here. I am really looking forward to having a great conversation with you on a lot of buzzwords, MLOps and DevSecOps. I am happy to be joining in a great discussion with you and talk about a lot of topics as well.

Jay: Cool. So to kick us off, tell us a little bit about what does it mean to be the director of continuous deployment of capability?

Hasan: Oh, it's a lot. What I am doing in here. So I am a technical director at the Software Engineering Institute under the Continuous Deployment of Capability group. As we name the continuous-deployment capabilities, our job and our mission is delivering capability to the end users. That capability includes any type of systems and software and using the modern techniques and modern practices, or any practices relevant to organizations. When I say the relevant organizations, every organization has its own business needs and its own complexity. Really, we would like to make sure that we are delivering capabilities [in a] timely [way], at high quality, and in a reliable and secure system that goes to the end users. We don't want to be going fast and failing fast, but we would like to go fast and deliver high quality for the end users.

That is my job, but it is really hard to do that. It requires a lot of knowledge, a lot of understanding of the business domain, a lot of connectivities as we are supporting various SEI customers including DoD and other industry folks as well.

Jay: On my side as interim director of AI for mission, our goals are kind of aligned. Our goal is to figure out how to build AI capability for actually implementing things for different missions. And as a similar thing, having to learn business domains and understanding how people want to use AI, and then figuring out how does it actually connect with their workflows. There is some good overlap in the types of work that we do. Let's talk about DevSecOps and how it relates to AI systems.

Hasan: All right. Yes, let's start from the foundation exactly. Maybe I should start from DevOps first, Jay. I mean there's a lot of people saying, *Why is DevSecOps? What is DevOps? What are the differences*? DevOps, it's basically a typical terminology putting *dev* and *ops* together, developers and operations teams. But in reality, DevOps is really connecting all stakeholders across the lifecycle. It really started based on the movement that came from the operational team because there was a disconnect between what the developers are delivering to the operational team, which is requiring infrastructure potentially, maybe reconfiguring their components, or maybe not ready for the changes as quickly as what the developers wanted. To solve the connectivity problems between developers and ops required other stakeholder involvement. That's the reason it's not just the *dev* and *ops*. It requires the business connection as you said, and Jay, we really would like to connect our business needs, which is our user needs.

When we look at the right, on the operational side, when we try to go to the left side in the lifecycle perspective, kind of a shift-left concept. To solve some operation problem, we have to do some homework in advance. We have to really talk about architecture. We have to talk about some requirements. We have to understand our infrastructure. We have to understand our deployment mechanisms to meet the user needs, the operational needs. That is the reason DevOps requires all stakeholders' involvement with strong collaborations and strong sharing and also using a lot of automation techniques because we would like to go fast. We cannot go fast if you don't use automation. We cannot go fast without testing, etcetera, as well. So it is a combination of sharing practices and also using the right tools and using automation with the goal of delivering a business capability that is kinder to the users. So it is a DevOps perspective.

Why are we saying *DevSecOps*? There was a long journey, but as a community, as a DevOps community, we either unintentionally or [because of] our timeline, we ignored security. We said *OK, we will do security later on*. We realized that since we are going so fast in the continuous delivery concept, CD, or continuous integration, the CI concept, we are using a lot of practices. All the continuous practices really make it very difficult to integrate security practices at the end when we are ready to release. When we are ready to deploy, the security became a bottleneck.

Now as a community we said, *Let's do DevSecOps because we have to address security throughout the lifecycle*. So if you have done a great job in a DevOps implementation, we should not call that a DevSecOps. But we are not doing a great job as a community, we are explicitly calling out that security should be part of lifecycle. That is what DevSecOps means.

When you look at this perspective, you always say we have to use a set of common practices. A set of common practices are sharing, communications, continuous integration, continuous delivery, continuous feedback, continuous monitoring. Maybe you say, *Jay, let's add continuous for everything*. Yes, I heard that, continuous of everything is basically what we are saying with DevSecOps. Using those practices and then using the infrastructures common to the DevSecOps pipeline, which is the environment that we are talking about. I think you have to really separate out the two elements. One is talking about the practices and how we are doing all this CD or CI and continuous, etcetera, then where we are using this practice is in an environment, which is our DevSecOps infrastructure, which is an end-to-end software pipeline or delivery pipeline. Basically we are building the capabilities and delivering from the beginning and through the pipeline. So that is overall the process pieces, the technology pieces, typical classical term, and then looking for common practices and common deployment techniques. That is what DevSecOps is right now in this day and age. It will change. Again, this is what we are using right now, but in the future, we may call it something else. But our intent is really good, fast in a reliable and secure system to be delivered like that.

Jay: I will say from past lives and past places where I have worked, making that transition is often hard at multiple levels. The initial thing of, you have got developers who are working in isolation, building up this thing, and then they try and throw it over a wall to an operations group, converting that to a place where handoffs are happening smoothly, people are having conversations about, *What is our capability both from the what is the software delivering, and what is the operations environment we're working into?*, has made things go a lot faster. It feels like all the talking should slow things down. But it turns out that it speeds it up because people have a better understanding of the context of what the application is supposed to do on the operations side, and what the capabilities of the operations side are, especially in protected environments.

Hasan: Jay, I am going to open up that topic. You said a key word, actually, I like what you said. When we look at operational needs, typically in an operational world, they are using a lot of common techniques already. Like IT folks are able to scale up the many deployments and many configuration elements for any system, any software they do. They already have a lot of mechanisms to handle the different configurations, the different deployments at scale. What is really missing are the connections. As developer teams and architects, we are not taking advantage of the knowledge that comes from the operational world. We are not able to take a knowledge that sharing what I have to do in advance, using the baseline images as an example, or using the same configuration that the IT operational person or team does, or using the approved software with respect to security that teams are using.

As a developer, we always like to get and write code. I have given a lot of examples from my class. I am teaching at CMU [Carnegie Mellon University]. Most of my students when I give a project to them, Jay, they really like to write the code. I say, *Hold on a second, I know you are going to write the code. You are just going to Google it and copy and paste and write the code. But there is a science behind it. The science is to understand what you are trying to build. What is the intent and what is the purpose? Then you can build up the software and systems*. *Still you are going to be copying and pasting. Still you are going to use a lot of existing frameworks. But as a developer or architect, we have to know what we are building, what is the intent.* We don't need to know everything, but at this we have to communicate with the team members, with the other stakeholders as we said, architects and testers, and operational team, security team. We can go much faster, much quicker, by talking to the right people, the right team members.

Jay: Definitely. That virtuous cycle of deploying and communicating back, this may be dating myself a little bit, but you talk about people who are moving from the cloud. I know there has been plenty of applications where I have moved the application over just lockstep not thinking about what the operational environment looks like, and how the cloud operates. That makes me create very large monolithic applications, which are very expensive to run in the cloud. Whereas once I understand that, *Hey, in this new environment, I should do things a little bit differently*. Getting that feedback from my operations team helped me rearchitect the application, so that it works smoothly and effectively and efficiently in the new environment. A lot of interesting work that is happening there, and the field has come a long, long way as we move from DevOps to DevSecOps. But on the horizon, we are seeing these MLOps, machine learning ops. Obviously AI has a lot of new capabilities that people are very interested in. What is your perspective on the relationship between MLOps and DevSecOps?

Hasan: I would like to talk about how MLOps term really started and what is the intent of MLOps, so maybe we can connect to some of the practices that we have been doing under the DevSecOps concept. So when we look at the machine-learning operation, we are dealing with the same type of problem that we discussed, Jay. We are building capabilities. Capabilities is X, black box. We don't think about the black box. We have running some software and some systems, and usually more likely has operational pieces only. As we start from the right side going to the left, from operational to the early lifecycle that we are building in software and systems, we start to see some problems, either we are not able to build up the software and system to meet user needs, which is business needs, or not able to connect the monitoring pieces of how our system is behaving in an application production environment.

I know that as developers or architects or data scientists we are thinking it is really great. Is that really great what we have built? Is it really working how it is supposed to in an operation environment? So the community said let's try to bring machine-learning team members, which are data scientists and machine-learning experts, to connect with their operational teams to monitor how the model is behaving and concur—look at the data sets as well for data sanitization, data cleansing, data preparations—to build up the model, then use that model in operational work as well. Because if we are building a model, if the model is not meeting the user needs as intended, then we are wasting our time, which is called a technical debt reality.

So we are spending a lot of time and money and energy building up a model, but the model a) may not be meeting the user needs, and b) may not be deploying the application context, which is creating a debt for us. So to solve that problem then as a community, we said let's bring the two worlds together, and call it MLOps, machine-learning operations.

So when you look at the DevSecOps pieces, the MLOps are using the same concepts of the DevSecOps concept that we have been talking about. Like what are the concepts? We said sharing first of all. Sharing means using the same business objective, same business goal. What that means for the MLOps concept, we would like to share our business objectives that we are building. We are creating a model for what type of problem we are solving. The problem usually comes from our business needs. Maybe we are looking for some fragile transaction, our transactions that we are getting in a lot of the financial systems. Maybe we are analyzing a lot of pictures. What type of pictures matter to us? Maybe we are looking for some audio records. It's more about the business driver, so really connecting the business drivers, then going back to our data scientists to say what the right data sets are that they need to be preparing. We don't want to be creating data randomly. We have to create data that meet our business needs, so we are using that data with the ML code together, which eventually we're going to talk about more. So we have the right data sets, then we have to really keep the data versioning as we are working on it. Why is this important? Because this is not a one-time job we do, Jay. It's continuous. Because I have never, ever seen that we are building an application that stays forever, especially with the ML systems that we are talking about. It is constantly evolving. We are getting better and better and better. I know we are all using a lot of daily MLOps applications in our life like some Siri, maybe other Google examples, many examples in our daily life. Even if we are using the face recognition on our phones or the fingerprints. All are getting better and better because it is using a lot of data. It is a continuous process. So what they are really telling us in a lifecycle perspective, what we think we are building right now is not static. It is dynamic. It is evolving. To meet those evolving needs in a lifecycle perspective, we have to connect our business objective, which we have to do. We have to measure it. Are we doing well? Is the model working as it is supposed to? So we have to collect the data from the production environment, feeding back with the new data sets potentially for data scientists. They can tune it. They can prep it. Now it goes to the machine, and they can spend more time to test different algorithms potentially, maybe different configurations.

When we look at this perspective, it is the same as in a DevSecOps component or DevOps environment as we have been talking [about]. Each of the configurations has some code in it. I know we have data, but we are configuring data different versions potentially. We have to know which version. The same concept is also true for application and build mechanisms. We are building applications. We are building software. Each instance of the software is a version. Each version has the code that goes together with the version. We are kind of labeling as a part of the CI process. Then also we are creating other artifacts with our application. Other artifacts are the environment as an example. Other artifacts are the test harnesses. Reflecting practices in AI/ML, and ones that we are building, we have to know what type of configuration parameters that we use in our data preparations and scripting. We keep the version of data, or we need to have a trace of the algorithms and some other techniques that we use in our machine-learning code pieces to keep the versioning.

Why does this matter to us? Because when it goes to the production environment, as the team of data scientists or the ML engineering team, we are working with maybe different versions. If there is something that we discover from the production environment, we have to go back to the previous version. How can we go back to the previous version and relate those requiring DevOps principles, such as using the shared environment, like using a repository-management system example. And also in other things we have to look at that environment perspective. How about if we test the model on different environments? It's going to behave differently potentially. We have to use a lot of DevOps principles again like infrastructure as code, which is IAC, which will help us to use the same configuration over and over. We can script those environments. We can run it and then use the new model. We can use the new data sets and monitor those feeding back to all stakeholders, the end users, data scientists, and ML engineers. That is how the DevOps principle is really helping to solve some challenges that we have been seeing in the AI communities.

There have been a lot of studies happening over there. Actually one of the studies, the 2020 State of Enterprise Machine Learning study, was clearly saying that more than 70 percent of applications are failing to deploy into the production environment, because either we don't have the right data sets or you are not able to deploy in a timely way to the end users, or you are unable to configure the environment properly. Or you are not able to really monitor how the system is behaving. These are the typical problems that DevSecOps is solving.

I give a lot of information, Jay? Maybe you can unpack a lot of content here, but it is just the overall concept is here. Let's go fast. Then you go fast. Let's be smart and look at all the data sets in our lifecycle perspective.

Jay: Absolutely. You said a lot of stuff that resonated with me to bring together. The use of static data sets, the ones that people can traditionally grab off of public sources, is kind of the equivalent of the copy-and-code that your students do from what they find in Stack Overflow. It's not really thinking about problem-specific issues or your specific business. It's looking at generic sets or good samples. A lot of folks who talk about AI now talk about AI as software 2.0. And in software 2.0, we code by example. That is what the data sets are. It is like bringing together lots of examples of all the different things that we expect to see in our data set, in our world, that we want to make predictions on. Creating new tools—and that is where we are right now—of how do you do code control for example? How do we make sure that we can figure out what data set was used for what model, under what parameters? How are we slicing up that data so that when someone sees a result in production, they can go back and re-create it by looking at, *OK how do we build that model? How can we augment that model then to see, does this new set of data help us correct for it?* Talking about things like data drift, as the world changes. Spam is an interesting case right? We develop tools for detecting spam using machine learning. What happens is we got spammers who got better at manipulating those rules and figuring out how to avoid detection again. That became maybe not a chicken and egg but more of a race of who can get further ahead with creating spam-creation rules or spam-detection rules. To your point of being able to monitor what is coming in, seeing what is happening, and then using that to communicate back to the team of, *Here are some examples that did not work the way that we expected them to*, really is critical for making improvements to the code, making improvements to the models, so that we can turn that around and get improved models back out there.

Hasan: Jay, I would like to add a couple of thoughts as well here. We need to also think about how to really share the information from operational usage on how the model is behaving and to feed back to the machine-learning scientists or data scientists or other stakeholders. The developers have to know as well what the model looks like as an example. To have that type of communication also requires a smooth operational environment, which is another point of view that is always missed. *Yes, we would like it to work. We can talk together. We have to share the same vision and same goal, but we have to use the same environments to communicate*. Because sending through email is different. Sending a chat message or the built message is different. And I tell you 99 percent, I always with my students as well, if I respond to email, nobody really returns my email. If I put some chat messages or a known Slack, I get a response right away. The same is also true. We learn we are in a society. We would like the same platform to communicate. Same platform for sharing information. That is kind of like the MLOps or DevSecOps pipeline that we are talking about. If you are able to collect the data from maybe test harnesses that we may have or the production environment we have, if it is not shared and stored in a common environment, in the pipeline, it is very difficult to connect the dots. It is becoming personal repositories. We don't want to have personal repositories anymore. We should have common repositories that we can share the information with all stakeholders.

Again, we are human. We always like our laptops. Sometimes I forgot what I did. I have to look at the history of what I am working on, which requires some sort of traceabilities or any type of information. If we track what the model is that we build, what the purpose is, what tricks we use, then we can go back and look at it from an improvement perspective. Otherwise, it is going to be a problem. I am sure it is the same as well for Jay. If I write the code about six months ago, if I look at again, I don't remember anything. It's a reality.

Jay: Absolutely.

Hasan: We are in a text-based environment, that's reality. The train is moving. And then always, our production environments are behind what we are working on. It is really telling us that we should have connected at both ends what is really working right now, what we are currently working on that can improve ourselves. Then we can go back to what we have done in the past, which is requiring a great end-to-end pipeline. In an engineering perspective, we can look at and trace that information so we can build up better.

Jay: I think one of the big things that changed for me in moving to DevOps and DevSecOps is the importance of making good commit messages. Really about being able to say, *What is this change doing?* And being able to communicate to my future self of like, *This is what I was thinking when I made this change,* so that when I have to go back and figure out why did I do this thing six months later, it becomes clear that this is what the thinking was. I think that is going to be one of the challenges moving forward with MLOps and AI is just, what is the right notation to make sure that we're keeping track of, *This is why this set of features was important*. Or *This is the set of things that went into these examples*, or *This data set that helps me figure out what was the thinking that made it make sense?* Definitely going to be an interesting challenge of communicating to my future self what I was thinking that went into this process.

Hasan: There is a study on that type of commit messages or the connecting with the environment perspective or the pipeline. A lot of organizations honestly are blocking the code commit if there is no case number associated with the commit. As a developer, yes, we would like to write the commit message into our code content. There is another enforcement on the code: why are we writing the code, what is the purpose? Tell me the case number, which is giving another lesson learned or the traceability for us, what things we are working on.

The same thing is also true for the ML perspective. What things are we working on? What is the version of data we are working on? What pieces have we produced? What is the purpose? Because we are thinking for a future as a set direction, we are planning our future work by recording ourselves. Actually I am going to share one study here, Jay, and I think I'm right. One of the big organizations, actually I think their name is GitLab, they build up a most knowledgeable organization as producing a lot of eBooks. What they have done, they recorded every lesson learned as they shared information about what they have done work-wise; they have built traceability They have more than 70,000 eBooks they can publish quickly.

So what this is really telling me as an organization, as evidence, [is that] we have to really collect information as shared environments. It is going to solve the problem as individual contributors become team contributors. If it is individual, I may not remember everything. Everything is going to be in my laptop, and then I will become a bottleneck.

You know the bus problem, right? If something happens to me, an organization will lose the infrastructures and knowledge of my work. What we are saying in an MLOps concept and DevOps concept, we would like to share the information with all stakeholders, keeping the same repositories—either mandating with the requirements on the code repositories that we committed or creating an organizational policy or practices. Let the people share knowledge with other people, either chat, or Wiki pages, or a SharePoint, something that can build up the knowledge. We can really use it over and over again because a lot of components are reusable. We can go back and take a look and improve better because we are dealing with speed. We are dealing with a lot of pressure from the end-user perspective. Let's get some feature out the door as quick as possible because we are racing with the market, either demanding the user needs that we have to serve as a business, or we have to really be on top of what we are delivering capabilities for our warfighters and the people that are in the field they can do their job as quick as possible, as high quality as possible, which we are racing with time.

It's not the same as before, where we have to wait a month. We cannot wait a month anymore, honestly. Sometimes, you have to wait a day. Like another thing, maybe I am kind of hijacking our conversation a little bit, but I have to give this example which happened recently. There was the Spring Framework where a vulnerability just came out on Friday. Now everybody is struggling in the community. *Am I vulnerable? Is it working in my machine? Is there any problem in my environment?* Now everybody is talking about the DevSecOps again, because of the traceability again, it is becoming about readiness. It is true for every application we are working on. It's about readiness. We cannot really wait another day or week to respond quickly in any outage or a business because the train is moving so fast.

Jay: Too many zero days are coming out where suddenly a vulnerability is discovered, and then people are taking advantage of it immediately. The time for the other side to take advantage of our lapses is becoming smaller. They are able to automate. SolarWinds told a lot of that story of automated platforms that are discovery of a zero-day vulnerability that was taken and used very quickly.

Hasan: Right. We should be making it more ready. We are talking about being ready, being resilient. Resilient means that we should be ready for any outage. It is also true for the data perspective as well. If the model is not working as it is supposed to be, how much quicker can we retrain that model based on the new needs that we have discovered and push it into the production environment? So we can really measure those metrics like MTTR or MTTD, mean time to recovery, mean time to detect. These are the key principles that are also true for the model perspective. If the model is behaving badly, how much more quickly can we work and respond to those outages immediately so the model will work as it is supposed to?

Jay: And how do we make sure that we are tracking the right metrics to do that? I think to your point earlier, you have a lifecycle where we are building these models for deploying them out there; the other part of that is making sure that we are monitoring them. How are we watching them and putting that back in that context so that all the stakeholders can be aware of, *Is our success rate staying the same? Is it falling? Are we seeing more false positives or more false negatives coming out of the model that we need to figure out? Is it doing the right thing or should we be replacing it?* That continuous monitoring of capability. We started that in DevSecOps to be able to monitor things. I think AI forces us to really double down on that lesson, making sure that all the good practices that we do for DevSecOps, we're really using for our new systems, AI systems as well.

I do want to take us in a slightly different direction. DevSecOps has been growing for the past number of years. I won't say that it is done, and it is fully mature, but it is definitely moved from a place where there were like thousands of options or many things to choose from to best practices or leading practices or leading tools that people know how to choose and have built good communities around them. In the MLOps side of things, we are still in our infancy. There seems to be every day someone turning around and providing a new framework or a new way of working. From your experience watching the growth of DevSecOps, what do you think is the best way for a practitioner to make sense of a rapidly changing environment in terms of the tools and practices that are out there?

Hasan: Great question, Jay. By learning from the history of engaging with DevSecOps for years, I always advise that people start from the *why*. It is really important. *Why I am going to build?* It is true regardless of whether we are building AI systems or building static applications, it doesn't really matter. It's always the why I'm going to build, then really build up the pipeline to serve our needs. We always jump to the wagon and build up the pipeline to serve the needs. We really have to change the equations. We have to build up the pipeline for our AI/ML deployments or system deployment based on our needs. It is important. We have to understand, why are we building? What is our target? What is our business environment? What is our technology?

We have to build a pipeline as a team. There are many existing MLOps pipelines available from different vendors, different cloud providers. There are many DevSecOps tools available in the market. Can one solve every problem? No, but really we have to look at our problems that we are looking to solve and then look at the tools to serve our needs. I give this analogy for my students as well. I always say if you go to the Home Depot, you are buying tools. You always have a project in your mind when you are buying the tools. You are not going to Home Depot just for the sake of buying tools just because you like them. That is not a true statement. We always have a project that we are looking for, trying to achieve this project. *I have this project. I am trying to achieve this project. What do I need to achieve this project? I am looking for the tools*. This is also true in the software world. Look at what we are trying to build. Then choose the tools. Then look at the environment again.

Again you are not doing this every time. Once you build up your pipeline with respect to your deployment delivery, then you have to monitor those and you have to maintain that infrastructure as well to serve your needs. I give another example here as well, which is very important. We always fall in a trap of letting the tools dictate to us what to do, which is the wrong purpose as well. We have to build up the common practices for our business needs. What are the common practices? We have to have a continuous integration, as an example. We have to have a sharing of the data model that we are working on. We have to have a habit of using the code repositories for the ML code perspective. These are the practices. These are the process. Then use those processes and bend the tools to serve our needs. We don’t want to use the tools to bend our process. It is too bent. If it is becoming a problem, then you cannot change your tools because the tools is dictating what to do. We should dictate to the tools what we have to do with our business needs, with our business objective, and with our common process. That is more of the structure that I am talking about. Otherwise, there are a lot of studies… Otherwise it will become a tool-junkie organization. There are a lot of surveys, it is happening. Again I am not against the tool, Jay. I am saying we have to pick the right tools for the right purpose. You know, like one tool cannot do everything, it’s impossible. But a combination of two tools that are serving our needs makes our work possible as well. Then maybe we can talk on another podcast about what type of criteria we should use to select the tools, you know some quality attributes and other capabilities.

What I would like to say overall is, start from *why*. Start from our business needs, and then build up what things you should have from a pipeline perspective, improve the pipeline. At the same time improve your set of practices, then the cultural elements come into the picture. You will see that as an organization, you have to reach out to many team members. Maybe you are going to talk with your data scientists. You will see some problems in your communication maybe. Maybe you will see some siloed organizations, who knows what you will discover? And start process-improving your organizational culture. At the same time build up the tooling infrastructure to serve your needs. And as an organization you will see which one is not working for you, which one is working for you, and then you will improve it. Improvement is always our business objective, which is what we have been saying since we started the podcast. It is important to serve our needs. That is solving the problem of why I'm building, to serve our user needs, our customer needs. What do I have to do and how am I going to do it, which is a typical process I always advise people.

Jay: That is great advice. I think starting with a *why are we doing this*, looking at it, can help people navigate what are the right tools and practices they need to adopt. It is not about following whoever is the trendy thing or what is the latest blog post that you read about. It is looking at your mission, understanding what you are trying to accomplish, and then figuring out a set of metrics to help you figure out which of these tools or processes are going to help you achieve your mission and get it going effectively. Obviously, a lot of things are going on right now, but I think that is a good heart of advice to go to: really understand what you are trying to do and heading in that direction.

Hasan: Exactly. As a community, Jay, we are failing on using the same configurations over and over again. No, we are not there yet. We may use the same tools or same code bases, which is growing in that direction. Everybody has different needs from an organizational perspective. Yes, we are using the same language potentially. We might be using the same libraries. That is perfectly fine. We are using a lot of dependencies. It is OK. We are using a lot of algorithms already. But implementation is really changing based on our user needs.

Jay: Yes. The changing environment, you said it earlier, everything is changing very rapidly now. Where in the past we might have months or weeks to deal with things, we are now getting close to days where the reality is changing. We need to make sure that we are adapting. To build fast organizations, having the right processes and tools in place can help us do that. Understanding the mission and then building the infrastructure to support that mission the best way possible.

Hasan: I am just going to add one more thing here Jay. I know we are running out of time, but I have to say that it is basically a balance of people and process and technology. You cannot have just the pure technology. You set up your great MLOps pipeline. It is working perfect with RStudio. Then you have all these gadgets and working with the container base—we're not going to solve the problem without the people to use it. Or you should have the process to let the people follow the right platform to use. It's a balance. You cannot build up one leg and then another two legs and another leg. We have to work as a team together as an organization, building up right processes, educate our people, and build up right technologies so we can really get better.

Organizational structures limit our success. We are always looking for improvement. For me there is no actual limit. *The sky's the limit* is what we have been saying at CMU for a long time. We have to improve ourselves. We will see a lot of problems as we are improving ourselves. As

I said, maintaining the libraries is a problem or maintaining some other data will be a problem. We may see some other scalability problem. We may see a data-storage problem. Maybe you are going to solve the resource problem, but we don't want to be getting everything in a great environment. Let's start to build as a kind of journey. We really start our journey. We know our vision, our end goal is to become an AI company. *Yes, we would like to become an AI company. We would like to go fast and quick. We would like to deliver our capabilities, which is our very high-level objective. How do we achieve our objectives? Let's go step by step.* Again, I am not saying that like a maturity level. I am talking about phased steps that we can build up as a team together. We can work as a team, and as a team, we will decide what is our next steps, kind of like using Agile methodologies as we go. Plan your sprints. See how much you can get done in a month, maybe in a sprint. Then you can get better in the next sprint, the next sprint, and learn what you have done in your mistakes. Maybe you have some communication problem that you discovered. Maybe you have some tooling problem you have discovered and fix that problem as you learn. Don't wait years to discover those problems. Learn, practice, and then feed back to the team. Improve yourself as an organization—that I can advise.

Jay: That is great advice. I think being able to think through the world of iterating and creating things. Try things out, give yourself a timeline, time boxing it, and then continuing forward is a great way to move forward. I think that's great advice. Any other parting thoughts you want to share with our audience?

Hasan: I am just going to finish up the one thing as you said. You said *iterative*. When I look at the static application versus the AI type of applications, AI is more iterative. Honestly, it requires a more iterative approach than a static application, because static applications we define those rules, we define the steps and functions in our application that we are building, and then let it go. But when we look at the data element, it is more iterative than ever before, which is really forcing us to be as iterative, as incremental as possible as we are building the model. I think this is a model that we should really think about. How can we improve ourselves as iteratively incremental, building up our models and using the data? It is really a key point. We should not ignore that. It is not a static application as we have built for years.

Jay: Absolutely. I think that is an important point to get through and make sure everyone remembers. This is a world where it is very dynamic, and we need to make sure that we're adapting to it. We are no longer in a place where things are static, but we are seeing a lot of change. DevSecOps and MLOps give us some tools to help us manage that change and help people with their applications deploying and succeeding in their missions.

Thanks again, Hasan, I look forward to having a continuation of this conversation in a couple of weeks as we talk about deploying AI systems with MLOps. For everyone else, thank you for talking with us today. We'll be including links to the transcripts, the resources mentioned during this podcast. Finally a reminder to our audience that our podcasts are available on SoundCloud, Stitcher, Apple Podcasts, and Google Podcasts, as well as on the SEI's YouTube channel. If you like what you hear and see today, please give us a thumbs up. Thanks again for joining us.