Episode 27: Robin Yeman
Episode Transcription

Mik Kersten:
Hello, and welcome to the Mik + One podcast, where I sit down with industry leaders to discuss the Project to Product movement. I'm Mik Kersten, founder and CEO of Tasktop, and best-selling author of Project to Product, How to Survive and Thrive in the Age of Digital Disruption with the Flow Framework.

Mik Kersten:
Before we begin, I want to take a moment to acknowledge International Women’s Day, which took place on March 8th. This year’s theme is ‘Choose to Challenge’, a call to action for challenging and calling out gender bias and inequality, seeking out and celebrating women’s achievements, and creating an inclusive world. I find this such an important cause, and am happy to say that since publishing Project to Product in 2018, I have dedicated all author proceeds to supporting women and minorities in technology.

Mik Kersten:
Many of the women whose work has inspired me most have already appeared on this podcast. Celebrating this International Women’s Day 2021, I am thrilled to welcome Robin Yemen, one of the foremost authorities on applying principles of flow and value streams to complex systems of systems. Robin is a Senior Fellow and Agile / DevSecOps Enterprise Coach at Lockheed Martin and her expertise spans over 25 years in software engineering with a focus on Agile/DevSecOps, building large complex solutions across multiple domains.

Mik Kersten:
Robin tell some amazing stories on this podcast, and I think will resonate with anyone who has tried to apply the concepts of Agile and DevOps at scale. So with that, let's get started.

Mik Kersten:
Hello, everyone, and welcome to the Project to Product podcast. I am so happy to be here with Robin Yeman, and Robin I met around 2015 while she was working on the Orion Spacecraft, which is somewhat infamous for being a way for us to get to Mars at some point. And I think even more interestingly is we're now entering this era where we're seeing microservices on planes and value streams in space. And I think Robin has been one of the people who has been spearheading the thought leadership, the way to approach working with these kinds of extremely complex systems of systems. So, Robin, welcome. It's so great to have you here.

Robin Yeman:
Thank you, I really appreciate it. You've got quite an all-star cast. So I'm a little bit humbled to be here.

Mik Kersten:
Well, it's been amazing to have people who've been applying the principles that you and I hold so near and dear around flow, Agile, DevOps and value streams. Some of what I've seen you doing and heard you talking about around applying it in this complex domain, where safety is critical, where regulations are critical, but where moving fast is more critical than it's ever been, has just been fascinating.

Mik Kersten:
And I do want to let everyone know that you're now actually working on a PhD thesis to help others understand the things that you've learned over the years. So, just before we get started on everything, can you tell us a bit about what's the title of your thesis.

Robin Yeman:
The title of my thesis is ‘How to Deploy Complex System of Systems Using Agile and DevSecOps’. And I know that's a mouthful. But all of the benefits that you get from Agile and DevOps on small applications actually are much larger, and even a bigger value at those large complex system of systems. It just requires some rethinking and refactoring to do that.

Mik Kersten:
And what fascinates me about what you're doing here in terms of these systems of systems, where you've got tons of software, mixed hardware and software, you've recognized the need of new ways of deploying software. I would love for you to tell us a little about Kubernetes running on a U-2. When I saw that cross my feet I was blown away. And then these extremely complex and critical life cycles. We're talking similar to commercial airliners, where you've got 50, 75-year long product life cycles.

Mik Kersten:
So, if you could just tell us a little bit about your journey through this because I think a lot of what we've known around this kind of product complexity, this kind of mixed hardware, software, and systems complexity, so much of it comes from industrial thinking, where lean originated, but I think the methods that you've been introducing and the new ways of working and thinking and end-to-end flow and value streams are very different than the way a lot of leaders in the space think about this. So, why don't you just start by telling us a bit about your journey how you got here.

Robin Yeman:
Okay. Yeah. I've been here at Lockheed for about 26 years. So quite a long time working on a variety of domains. I started out with submarines and displays and had the opportunity to work on Aegis, which is a large combat system, moving into supporting things like F-16 and F-22. And then on to, as you said, when we tagged up, I was supporting Orion during their journey. So all large systems, but completely different domains with completely different backgrounds.

Robin Yeman:
Now, one of the things that is most visible that I see when we're working on these large systems is almost the org structure in the value stream. So I know that it's been said many times, but one of the problems that we pretty much see on a regular basis isn't even just the fact that we do have these different functional organizations. It's that the organizations use different language that actually evolve from the same type of thing.

Robin Yeman:

So if you look at program management, they focus heavily on lean and lean startup, which is great and minimum viable products, we're really working towards those. But when I talk to folks in systems engineering, they're heavily focused on systems thinking, but not really realizing its inner relationship with lean.

Robin Yeman:
And then we move into systems design. And the design groups really focused on design thinking. You'll see a number of workshops around the organization on design thinking, again, looking at how to evolve a solution collaboratively with cross-functional perceptions. You get into the hardware, and they talk about rapid prototyping, really harkens back to a lot of this stuff that you've seen in industrial manufacturing, and not realizing that that rapid prototyping really is pretty much close to their brethren in software, which is Agile development.

Robin Yeman:
So, they're using that type of language. When I speak to test, they're like, "Yes, we really need to get into shift-left testing." They're all talking about shift-left, but they don't realize about the rapid prototyping or the design thinking. And then when we get out to operations, you're starting to look at ITIL infrastructure management and technology business management.

Robin Yeman:
All of these things want to deliver capabilities at the speed of relevance and optimize the delivery. But each group has a different language to discuss it. And they don't realize that they're actually saying the same thing, which causes huge delays. This is before I even get into any of the technology-type handoffs that you're going to see.

Mik Kersten:
Yeah. So that's fascinating, the way you present it. Everyone is around the same goal, which is moving faster, delivering more relevant things to markets, and in this case, to missions and to systems, faster. So, you do see them speaking the same language, though. This is in the end, all embodying leans, presumably small batches, iteration, flow, feedback, and such.

Robin Yeman:
Well, what I see is, they each have an intent. And if you were to abstract it up a level, I believe it's absolutely one and the same. But the people within the organizations have unique language, I don't believe they're speaking the same language. So, when ITIL talks about the different types of feedback loops, they think that's a completely separate thing than what Agile is referring to.

Mik Kersten:
So that's really interesting. Because thinking flow, thinking value streams, we know it's one feedback loop that has to cross these silos. But I think this is the common thinking and the common way of deploying design thinking; that design thinking is deployed in design.

Robin Yeman:
Absolutely.

Mik Kersten:
And you've come at this largely from - and in terms of the way that you've helped Lockheed and some of these programs transform - largely from the Agile and DevOps side of things.

Robin Yeman:
Yes. But I guess the interesting thing about me, and you could probably tell, is I'm hyper. So I have not been in one organization. And I don't know that I would have seen it if I had. So, during my tenure, I have been a program manager, I have been a test director, I have been a software engineer, a chief engineer, I have been primarily working in operations.

Robin Yeman:
So, the fact that I got to go visit different areas because I like to see different parts of the system clued me into the fact that we're actually really talking about the same thing, we just don't realize it.

Mik Kersten:
Can you just give us a little bit more depth because this really is a fastening perspective. I've seen bits and pieces of this, but the visual that you paint of all these different groups trying so hard to move faster, speaking different languages, and only establishing a flow and feedback loop within their silo.

Mik Kersten:
Whether it's the rapid prototyping or the design, I'm imagining that what you're seeing is that it's very difficult to do design thinking, where you don't have a sense of the software of the capabilities of what you can actually deploy, what kind of user experiences you can even deliver on, what kinds of devices while the team is rapidly prototyping something completely different in terms of a heads up display or something else in another group.

Mik Kersten:
So can you give us just some examples of this because I think this is such a powerful visual and so indicative of what we're seeing across federal and commercial organizations.

Robin Yeman:
Absolutely. So, for example, I was working with one of the teams and I tried to explain to them Agile for hardware. This was years ago, I want to say probably 2007, 2008. And they're like, "No. We know that Agile is for software. It's not used for anything else." And I was like, "Okay, let's step back. Tell me what it is that you're going to do."

Robin Yeman:
So one of the things they had to do is to identify if they could put racks in space that you could put hand gloves in there. And I was like, "Okay, so how are you going to do that?" And they said, "Well, that's easy. We're developing a cardboard prototype. And we created it the exact dimensions and we put our hands in there." And I'm like, "Oh, so rapid fast feedback exactly if I was building software and writing a quick unit test and trying it out." I said, "So the domain and

the material is different. But the thought process is exactly the same." And I think we get so engrained into that thought process, we can't see how it would apply to other things.

Robin Yeman:
The same thing in management. We talk about things like story mapping in Agile all the time. But if you go back to the 40s when they started program management-type discussions, precedence diagramming. Well, precedence diagramming looks a whole lot like story mapping - different language, same thing. So if I want to - and this is usually how I am able to work with our programs to make these large leaps, because to them, this is scary stuff - is to say, "Actually, I'm really just talking about precedence diagramming here." "Oh, that's okay. We can do that."

Robin Yeman:
But the problem is, if you're not aware of that language, if you haven't hopped around and seen these different types, you don't realize we're actually saying the same thing. And I think what you see is people grow up and they become managers, and they're program managers, that's their number one job, they do that forever. So they don't actually ever see that in operations, this infrastructure technology, ITIL, is really looking at a lot of the same things and operations that they're doing when they're starting off their programs.

Robin Yeman:
And so, because we have all of these amazing different capabilities and processes, we're not coming back to that lean value stream where they all are. They're all saying, "Hey, let's limit work in progress. Let's really visualize our work from end-to-end. Let's create a connection between 'once I create my models, how do they flow into the software? And then once I create the software, how do I validate that it actually implemented the model?' So what does that test look like?"

Robin Yeman:
And primarily in order to deliver a solution, my customers care about solution. They care about when I'm going to deploy a satellite, when I'm going to provide the next missile capability. They don't actually, even though they may think they do, care about how many lines of code I wrote, or even how fast I wrote them. Because if I wrote them really fast, but the test team was off doing something else for another team, the customer doesn't actually get to realize that speed of delivery.

Robin Yeman:
So, until we can really develop that Rosetta Stone, that common language that we could use across so that people in operations know exactly what somebody in systems engineering is talking to and can relate it to something, it makes it very difficult to get products and services out the door rapidly.

Mik Kersten:
Yeah. Ever since we met five years ago, I think you and I have had this very similar perspective and been on the same journey of helping organizations develop that common language. The

thing that I learned through this, and I think that you've got some amazing examples and insights on, is that some of these languages, they're there for a reason. The precedence diagramming, that evolved from something, it became its own language, people started using it. It's the same with story points, they became their own language; a cardboard prototyping is its own language.

Mik Kersten:
And as I've worked more closely with different silos and different organizations, I've actually developed this appreciation for the languages that they've developed. And the way that they vary. The way that ITIL teams operate services versus the way that SREs are working right now versus all of the technical language and jargon that we see around modern cloud architectures. These things are really important languages for those teams to do what they do well, but there's just such, I guess the challenges is, it's the Tower of Babel. We need some Rosetta Stone.

Robin Yeman:
Yeah. A translation at this layer, not that they can't specialize, but a translation so that they know that they're not redoing the same thing. When I first started working Agile programs, and that was about 2002. Here at Lockheed, we always get a requirements database, whether that's in DOORS or whether I get something from a requirements document, but the first thing I did was I decomposed all of those different requirements in DOORS.

Robin Yeman:
Then I was like, "Okay, so we're doing Agile now, we're going to have an Agile product backlog." So I created the Agile product backlog: epics, features, user stories. The first thing I ran into is a spaghetti mess because I had done all of the decomposition in requirements and now I'm doing all of the decomposition in scope and functionality.

Robin Yeman:
Well, decomposition is decomposition. Pick a place, whether it be in your requirements tool, or your product backlog, and do it once. So, from then on, after I managed that, it was quite convoluted. I typically leave requirements pretty large within the requirements database and do my decomposition in the backlog so that I can link across a requirement or a grouping of requirements directly to an epic and have one layer of decomposition.

Robin Yeman:
In the past, people thought of software engineers doing Agile, systems engineers doing systems thinking. They're both doing decomposition, they're using different terminology with that decomposition is, and we're causing double work. And not even just double work, we're making it very difficult to even interconnect with one another because of that double work.

Robin Yeman:
The same thing happens in modeling. So we build these model-based systems engineers or model-based prototypes, things of that nature. But often, they're not connected directly to the software and the technology. So, what's implemented doesn't actually come back and validate the model. The models get out of sync, but the models are the things that are used for things

like design reviews. But now, the map doesn't match the terrain anymore. And so, we try to make adjustments to, let's say, the software and the capabilities, based upon what we see in the map. But because they're not really the same anymore, again, we cause additional rework.

Mik Kersten:
Yep.

Robin Yeman:
I see it all the time too.

Mik Kersten:
Yeah. And so, I'm just reflecting on my own journey with this because I think there are two key things you've said. I just started thinking that if we could just give everyone their own languages, and somehow federate those, synchronize them and connect to different silos, that would be enough. But I think as you're noticing, it's helpful, we've done some things together that have actually produced those results of connecting DOORS to product management and Agile tools, it helps.

Mik Kersten:
But if you do it the wrong way - and you just shared a really important insight - I've worked with a number of organizations where at a program requirements level, they actually need to decide a feature may be implemented in hardware or software. And until you get little further down the experimentation or MVP or design thinking side, you don't even know.

Mik Kersten:
It's just such a great example that you're giving, you now feed out those requirements to all those teams who do all that duplicate work, none of it flows back to actually make a sound decision on where we should do this, how we should do it. It's this tremendous amount of waste and duplicate work.

Mik Kersten:
So, you're saying I think something that's really important, which is to understand where that decomposition of work needs to happen. And then I guess, in the end, also, to connect it back. You might learn something in that. The tricky thing I'm seeing as well as step one is what you said. And then you might learn something about the composition. We'll never get that performance profile in software, we actually need an FPGA here, or whatever the learning might be.

Robin Yeman:
Exactly. And if we don't have that feedback loop… So I always figured the ideal (now, this is just my ideal) is that you link your requirements database, which I think is the ‘ask’, the ‘what do I want’? Two, your product backlog. That is the ‘scope’, that's the ‘how I think I'm going to do the work’.

Robin Yeman:

Linking to the repository that you're going to put that work into, whether that's a git repository, or whether it's a document repository, whatever I'm building. And then linking that over to my automated test suite, which is then going to come back and feedback into my backlog because that's against the scope, which then will trace back into that originating requirement.

Robin Yeman:
And I think if you link all of those tools and create some sort of mapping of the language, people would have that common understanding of the vision. You could remove a lot of that duplicate rework, which we do over and over again, and also remove a lot of the defects because of that duplicate work. A lot of times, it's not some complex algorithm that's the problem, it's down at the basics.

Mik Kersten:
Yeah, exactly. I could not agree more. But I do think that your approach of thinking about where the work is decomposed, is actually a really important way to look at this problem.

Mik Kersten:
And then, the other thing I've noticed over the more recent years, is back to your point on the common language. So, I actually thought we could solve this; connecting these systems, these different vocabularies, these different ways that people work and the way that they collaborate, they plan, they prototype, and they build systems.

Mik Kersten:
But then what I realized is that that wasn't enough. Because we ended up seeing these organizations over and over who seem to be doing all the right things and followed some of what you just said. But where they were ending up was that in the end, it was all these local optimizations within silos, where the Agile team was doing great, the design team was doing great, the hardware team was producing amazing things that executives were just thrilled to see.

Mik Kersten:
But like you said, the end-to-end time of delivery was unknown. And it just felt too slow. It was not being optimized. It was just what we now know are (or for manufacturing), it's the cycle times within each of those silos that you described that were being optimized.

Mik Kersten:
The whole point of the Flow Framework and the Flow Metrics is that we need to establish a minimal common vocabulary. Those languages in each of the silos are important but we need to think about not just the cycle time of how long it took us to prototype, or to release some software and push some updates, but we always need to have a separate language that's oriented about how the value we've delivered. So flow time, the time to value, is something that everyone needs to understand.

Mik Kersten:

How have you been approaching this? Because it's how you think, it's how you've inspired me to think. When we're trying to get people to think of these end-to-end value streams of what was that line you said - you had that great line, ‘delivering value at the speed of relevance?’

Robin Yeman:
The speed of relevance is this: So potentially, if I'm Netflix (I’m not) but with Netflix, there are updates every second. And that's perfect because that's what they need. But let's say I want to make a deployment to a weapon system like Aegis, well, there's actually a lot of different things that I need to do to make that update.

Robin Yeman:
That being said, I've also got multiple different chips with different hardware on them. So, speed of relevance for Aegis could be an update every three to six months because I've got to train people on that new capability, I have to update a series of documents, there's a huge amount of work for that deployment.

Robin Yeman:
So speed of relevance is as fast as your customer can consume and that is valuable to them. Versus everybody's like, "We need continuous deployment." Well, for some of my customers, that would cause an overload. And actually, they wouldn't be able to close on their missions because they were spending all of their time trying to learn the new products that I was pushing to them.

Mik Kersten:
Yeah, exactly. We’ve had this obsession in our industry around deploys per day and these kinds of metrics because they worked for a certain class of complex systems, that some of which you just described. I think the whole point of thinking through product value streams is you define - and I think this is a better term now that I use in my book – the speed of relevance in the Flow Metrics - the Flow Time - for each value stream. That's what you go for.

Mik Kersten:
And if you're below that, that's great.

Robin Yeman:
[crosstalk 00:23:14] increased, yeah.

Mik Kersten:
Yeah, and that can be. The examples I've given is, we've got extremely innovative organizations. Some of the tech giants, they have these large SAP systems underneath still. The speed of relevance on those systems is quite a bit slower than their mean time to repair if something goes out on the public APIs.

Mik Kersten:
So, even in the most innovative organizations we think of, each value stream hasn't understood the speed of relevance. And if it's getting too long, then you look where to shorten it.

Robin Yeman:
But if you want to speed it up, like the natural inclination is just to speed it up. So you're adding resources, you're adding waste because you're putting more resources to speed it up to actually sit it on a countertop somewhere before somebody can actually use it.

Robin Yeman:
So coming back to Don Reinertsen's ‘Principles of Product Development Flow’ that load equals the delivery speed there - don't go faster than you have to because you're just spending more money and more time, but you're not getting more value.

Mik Kersten:
Yeah, exactly. I think what we've seen happen across organizations is that the amount of investment in Agile teams in many places are more advanced than a lot of investment in DevOps and release automation, but the wait states are no longer there. Yet the investment continues to help those teams go faster, whereas the bottleneck is upstream of them or downstream and compliance or security, or something of that sort.

Robin Yeman:
Absolutely.

Mik Kersten:
So, it's the start of 2021. How I've been trying to help organizations is just to make that visible. That's it. Just to make the Flow Time visible, to make the WIP visible where it's queuing up, to make the wait states visible and the like through the Flow Load and such with my teams. You're actually helping these organizations improve. Whereas the goal of the Flow Framework is just to have this defined, measurable, and have brilliant people like yourself structure the improvements, the experiments, the things that will actually get that speed of relevance down to what it needs to be.

Mik Kersten:
How are you approaching helping these programs actually have this perspective? Because one of the biggest problems I continue seeing is exactly the opposite of the way Donald Reinertsen wants us to see the world, which is end-to-end customer-centric lifecycle profitability. How are you helping change this thinking? Are you getting people to thinking in common lean terms because, in the end, these go back to lean principles?

Mik Kersten:
But when you engage with these teams, let's just assume, all of a sudden, you can get them to measure where the bottlenecks are, it looks like the bottleneck is in this duplicate work that's being done between hardware and software, because they both decompose, that same thing. How do you get people to change their thinking? And is it easy?

Robin Yeman:

Oh, is it easy? I think we know that it's not. I think the first thing is, is people have developed a mental model-based upon what they've learned in their careers or their lives. So, you can't just change that overnight. The easiest way that I've found to make change, and to convince people, is to actually use their language.

Robin Yeman:
Because honestly, I was a software developer. But if you were to say, ‘was I the best software developer you ever had?’ No, there were better software developers. I was always like, "Wow, I wish I could code that fast." Or if you look at, ‘was I the most proficient tester?’ I'm pretty good at a lot of things but I was never awesome at one thing. But the one thing I have been able to bring to the table is by visiting these different areas, by being interested in how all of the system works, I have been able to learn the different lingo from management versus systems engineering versus talking to the hardware engineers versus talking to electronics and firmware.

Robin Yeman:
And when I'm talking to somebody, I use their language. So I internalize what the core concept is, and use in their language. When we talk about things like that, people want to know, in management, at least in a lot of our government programs, what's critical path?

Robin Yeman:
Well, critical path is the soonest that I can get something done. So we're just laying out dependencies. And systems engineers really understand the pain of dependencies. So that's how you're going to communicate with them. If I'm talking to test, it's pretty easy to say, beginning with the end in mind actually reduces the number of tests you have to create. If I begin with what the outcome is, the answer to the test question that I can get there.

Robin Yeman:
So. I think part of the benefit I bring is actually being able to see this cross-view and almost be a translator, because a lot of times, people are scared. The first thing they would say is, "Robin, we build safety-critical systems. Agile, that's just fast development.” And when you walk them through what Agile really is, which is basically empirical planning, the scientific method, evaluation from my hypothesis, trying something out, and validating it back and iterating, they get that.

Robin Yeman:
We try to make things sound too tough, too complex, which then makes it sound like that's going to be a huge uphill battle. And that has allowed me to be really successful. You can't tell people, “it has to go this way because I said so”. You can't tell them, "Hey, just listen to me because I know what I'm talking about." None of that works. You have to appeal to where they are and their mental model. And let them visualize it in their terms.

Mik Kersten:
Yeah. I think there's something so profound to that. Helping people use their language. It gets them bought in, and it helps them understand that these are the same concepts. And I've noticed this interesting effect when you start doing that. Because once you start using their

language, you can start saying, "Okay, well, where are you getting stuck? Where is your bottleneck?" And then, all of a sudden, you're saying that to someone putting out an MVP. It's like, well, the software is not moving fast enough.

Mik Kersten:
Well, you need flow between you and the software teams or the prototyping teams, we're still waiting on them. Meanwhile, the hardware teams as well, we're still waiting on the design, the specification, the business, the analysis that we need for this and we've been waiting for four weeks, and the meeting got rescheduled three times.

Mik Kersten:
So, there's this natural pull I find each of the silos has, because they all want to deliver value faster and they have upstream and downstream dependencies. And if we can actually show them where things get stuck, where queues are too big, and where you've got these long wait states, in my experience it's pretty amazing how quick the buy-in to this more end-to-end [crosstalk 00:30:26] is.

Robin Yeman:
Oh, yeah, definitely. So, I can tell you about this old case. So I want to say probably in 2005, 2006, I was working this mission-critical program, but primarily for sustainment and operations. And like I said, I am very curious, very nosy, I want to see how things work.

Robin Yeman:
My primary job is to develop these patches, and then they would deploy these patches into operations to make certain things happen. I created this patch, and the first thing I did is run down to what they call the racetrack to visualize it, see what happened in operations. But when I got there, I saw an engineer updating my patch. I was like, "What are you doing? Why are you doing that?"

Robin Yeman:
He said, "Oh, I just have to make it work in operations." Oh. So, we know that that's a simple environment statement at the front to say where I'm at, and setting my paths. So he's just updating the paths, which he says, "Doesn't take very long, maybe 5, 10 minutes." But keep in mind that I was an engineer within a cube of 400 engineers whose entire jobs were building patches. And there was an entire set of teams down in operations that would update our patches to work in operations.

Robin Yeman:
I gave it a try. I was like, "Oh, this is what you need?" And I went back, and I started putting that at the front of anything I did at the front of all my code. And people were like, "Oh, she's really fast. Her patch is always going fast. She must be an amazing developer." Like I said, I was never the best developer. I was good. It had nothing to do with that. It's because I knew what was in the next step in that value stream and it resulted in my patches going in, I want to say 75% faster.

Robin Yeman:
People would come to me specifically because they're like, "Robin's will go in first." And the reason Robin's will go in first is because they didn't need somebody to just make a quick update to make it work in operations. Now, anybody, three tiers up would never see that. They would not realize that. But the amount of delay that just something like that costs is amazing. And so, by simply doing nothing other than being curious to see what the next person had to do with my product, changed the game.

Robin Yeman:
I see that time and time again, across the board, when I'm working on models. “What are you going to do with this model? How are you going to put it into the digital twin? Oh, well, if you have to do that, and this is how I'm going to set it up”. It changes how you build if you know how somebody else is going to use it.

Mik Kersten:
That's amazing. So, this amazing level of systems of systems thinking that you've achieved all came from you selfishly wanting to get your patches over faster?

Robin Yeman:
Yeah. I'm nosy, people will tell you. I'm curious. I want to know what's going on. I'm the first person to ask why.

Mik Kersten:
Yeah. And that's exactly what we need to, is to elevate the people who can think that way, who realize, okay, just with 2% extra effort of mine here, the flow and time to value will be reduced tremendously.

Mik Kersten:
Thing about systems is that we need to make these things visible systemically as well, because we can't rely on, and there were another 399 people who had not come to this conclusion as quickly as you had. And this goes back to this thing that you've told me repeatedly that's been haunting me since you told me. You said to me, we know these things. The people listening to this podcast are going to just be nodding their head as you're saying some of these things.

Mik Kersten:
To quote you directly is, "If everybody is saying it, why aren't we doing it? So eloquently stated, why this is so important, is just one small example. And if you pile on these kinds of improvements and look for these constraints, you can bring complex systems of systems to market 5 or 10 times faster. And that's a really profound statement and with a profound impact.

Mik Kersten:
I know this research question has been on your mind with you working on your thesis, if everybody's saying it, why aren't we doing it? Give us more color just on that question itself.

Robin Yeman:

Anybody who's been working on their dissertation, they know you come up with your top 10 questions, and then you try to narrow them down. And it's been really difficult because you're supposed to develop new knowledge, new material that hasn't been developed before. And the more research I do, the more I see that these cool ideas are everywhere.

Robin Yeman:
I have done a lot of research and reading on digital engineering, a number of papers on model-based engineering, Agile, DevOps, DevSecOps across the board. And they are all saying the same things, which then led me to go back in and because I work as a government contractor, the first thing I did is is say, "Okay, so have we always known this in the government?" And I went back a few years to read different defense innovation board papers, which is a commercial board to bring innovation to the government.

Robin Yeman:
And the interesting thing is there was a paper that came out in 2018 from the defense innovation board saying we need to change how we build software. And there was actually a very similar paper that said the same thing in early 2000s. And then, another very similar paper at the end of the 1980s. And so, in my head, I was thinking, we keep saying the same thing. It looks like even across the board, people are coming to the same conclusions. Why aren't people implementing them?

Robin Yeman:
And I think it's too easy to say, oh, they're being stubborn or it's change, or it's hard. There's something there. I don't know what it is. But there's something there that's causing us not to buy in and make these fundamental changes that would make our lives easier and allow us to deliver products and services faster.

Mik Kersten:
Yeah. For me, there's a really similar question. Well, first of all, my perspective was some teams have mastered the right ways of doing it in incredible ways. They're able to bring more value to market than an entire organization. For me, that was a very visceral thing because I learned it through open source. It was mind-blowing to me that it was coming from open source.

Mik Kersten:
And then in my thesis, studying large organizations, software organizations, and I was doing the productivity masters like, is it really true that this team of six loosely collected individuals working on Tomcat or Kubernetes, or something else, that this open-source project can deliver more usable value to market than an organization of 1,000 people, like this can't be true. But with the right kind of systems thinking, you start realizing, well, when you stack all those impediments and that slowness, and you fix all those things, you really can get to 10x, 100x, in some cases, 1000x to levels of flow of value.

Mik Kersten:
And then like you, in studying... For me, it was when I was writing the Project to Product, I was like, okay, I just kept realizing that I needed to read books I hadn't read. I skimmed, I looked at

presentation on whatever I'd done. So I end up reading what must have been a couple of dozen books over the course of writing the book. And I had the same vision as you. These are fundamentally saying the same thing. And then, somehow, interestingly, they actually all do point back to Don Reinertsen. But that's another story.

Mik Kersten:
And, of course, his work builds on lean and Gene can point me at re-engineer the corporation. There's the same core here. So I think your research question is the question I think we should all be asking ourselves as we're trying to help our organizations in 2021 is, given we know the practices… And this is a question I posted to Don on the panel that we did together with Dean and Gene on the Global SAFe Summit, is like, [inaudible 00:38:46] actually gave us the practices many years ago now, maybe the bottleneck really is us applying those.

Mik Kersten:
Robin, you're saying some key things that I think I want us to understand better and to take away is that you're somehow implying that the lack of this common language, it's somehow related to the fact that the practices are known, they're being practiced in silos, but they're not being practiced end-to-end. So do you think that's one of the reasons that there are no common language? Is that one of the things that your research question is going to unveil?

Robin Yeman:
Well, it's one of the things that I'm going to research. I don't know. But from an anecdotal visual, I've seen us do the same thing, like decomposition in multiple places because we didn't realize somebody else was doing that as well. And we had called it a different name.

Robin Yeman:
I was chatting with a new program. They're bringing on a consultation for a consultant company to do Agile management for the program. And I said, "Well, we're already implementing Agile. You can, but do you need a separate consultant to do the Agile management?" And they are like, "Oh, you're working on the technical, Robin. We need to understand how you manage the program." Which comes back to fundamentally, they don't realize that actually Scrum, entirely a management methodology. It happens time and time again.

Robin Yeman:
Another funny story was over in the UK, they really wanted to get their entire team ready for being able to implement Agile. They trained all of the program managers on DSDM. They trained all of the teams on Scrum. Again, we just know they're two different flavors of Agile. But if I'm in program management versus software engineering, I think that Scrum is the way to go for software. But over in the UK, they hear a lot about DSDM in management. It's a language thing and not necessarily a step back to be able to make sure we're on the same page.

Robin Yeman:
Because if we're speaking different languages, it's just like that canoe. I've gone canoeing with my family, and they always do it wrong. I'm doing it right, obviously. So, when you row at a different speed and a different way, there's one thing that happens, you just go around in the

circle. Unless we're all going in the same direction, at the same cadence. And so, if we can't create that common language, I feel like we're spending a lot of time in the circle before we actually get to the goal.

Mik Kersten:
Yep. And obviously, even if that cadence is slower than your cadence, Robin, goes straighter and faster. So, I think the interesting thing is that we are seeing some others doing it, and doing things the right way. Your story of these patches. Your recent story on the Kubernetes on U-2.

Mik Kersten:
So how do you find those examples of these things working again? Because my challenge is when organizations and leaders within organizations, and different functions within organizations, make all these excuses that, "No, that won't work here." And then, of course, you break through that saying, "Oh, well, the language you're actually using is the language of Agile, is the language of lean. And I think your unique approach is that I'm going to stop forcing-

Robin Yeman:
My language.

Mik Kersten:
... Agile down your throat, I'll actually adapt to the language that you're using, which I think is much more successful than forcing Agile and DevOps terms down people's throats. I've certainly experienced failed attempts of me trying to do that.

Mik Kersten:
So, tell us a bit about the successful examples that you've seen, where it has worked, where it has broken through a barrier.

Robin Yeman:
Oh, it's been amazing. Again, it wasn't my fete, but the aeronautics team put together Kubernetes clusters for the U-2. It came out of one program manager saying, "Well, what if?" Because typically, people are like, "No, we already know how this works, just leave it alone."

Robin Yeman:
And by doing that, one of the things that they realized is the deployment was just so much faster. They could make changes and get a deployment out, which we all know, but that's new for them. Processing in data power because of the cluster was such that they'd never had that much data before, which allow them to do different things with the mission than they'd ever thought of. And the program manager told me, he goes, it's like breathing new life into the U-2 because a lot of these programs, a lot of these large complex systems are not brand new. They've been around for 20, 30 years.

Robin Yeman:
So being able to take something and breathe new life into it, and get new value out of it, is huge. The only thing that we can do is say 'what if' and keep working with the next program. Could we

do this? And if you can't, why not? Ask the question. And once you start asking the question, I think you see amazing results.

Robin Yeman:
We've had a number of our IRADS doing the same thing. You're like, oh, actually, you know what, it used to take us, and it did, no kidding, months and months to get an environment out for this brand-new team. I had another team get one in a matter of hours. They were like, "How is that even possible?" To them, it was magical. Like, is that even possible?

Robin Yeman:
They had thought of, "Well, I need to purchase hardware, and then I need to image the hardware. And then I need to put the different tools on it. And I need to integrate it." This all takes time as opposed to, “I had an AWS image that you can use today on somebody else's hardware.” It's not rocket science to us. But when people see it and experience it, it's amazing. It's just allowed to deliver amazing.

Mik Kersten:
So, what if, and then those successes. Because the other thing that you're saying is the people who have trouble seeing this, that there's a different better way that's this transformative and that allows a completely different space of outcomes. So, the guidance you're giving us is to do the what if, which will then cause new ways of looking and working and focusing on the value. And then, I guess in the end, showcase the outcome of this new way of doing it.

Robin Yeman (45:04):
Showcase those short-term wins. That always creates a huge amount of momentum. Me and you have talked a little bit about the different software factories in the DOD space that the Air Force really pushed. And in 2018, you had Kessel Run, you had one of those software factories. There's about 30 to 40. To me, they feel like popcorn poppers. They're popping up all over the place where they're building mission-critical capabilities, and the knowledge of what they were able to achieve on Kessel Run and being able to share that really resulted in that momentum of being able to get those others.

Robin Yeman:
So, showcasing those short-term wins is huge. Even if you, yourself, and I found this myself, don't believe that it was a huge win. Sharing that with others who are not used to thinking this way, all of a sudden, creates a train in itself.

Mik Kersten:
Yeah. I think that's so key. And Bryon Kroger, one of the visionaries of Kessel Run, in a previous edition of this podcast, he went through that. He went through how they thought about that very similar to your U-2 story, continued capabilities. And then just the power of showcasing that.

Mik Kersten:

So I think it is, as people look into their plans for this year: Create, figure out what that ‘what if’ is, and figure out what that showcase is. Because in the end, we have to break through people not doing it. They're not having the right organizational structure in place, structure and managerial program structure. And I think you're giving us a pattern for doing that.

Mik Kersten:
So Robin, the other thing I'm taking from this is as that happens, as you've done the ‘what if’, you've created the existence proof, the proof point. We can do this. I think that the U-2 example is so extreme. So if you can actually prove that on the U-2, then anyone listening can probably prove that on their own masses of legacy and older systems and the different things that they're struggling with.

Mik Kersten:
The other thing that I think is so valuable that I've seen working on these things, is showcasing that with a common language and a language that resonates with the people that you need to impact, to change the structure and dynamics of the organization. So, to say we brought this to market this much faster and by actually given this team the space to build this and reducing their flow load, reducing their WIP, they were able to innovate faster than we've ever seen in this organization. By allowing them to use modern web and cloud technologies like the AWS [inaudible 00:47:38] serverless, they're showing us how we can scale this within our organization.

Mik Kersten:
So I think the other thing I'm hearing is that just to make sure to present that in not the language of ‘we were able to stand up the Kubernetes cluster that much more quickly’, but ‘we were able to deliver value at the speed of relevance’. And that's what you've been asking for all this time. And let's scale that out in the back half of 2021.

Robin Yeman:
Absolutely. Don't focus so much on the inputs, which I know I'm not the inventor of the statement. Focus on the outcomes that you're able to achieve. And by using that language with those stakeholders, I think you can accomplish so much more than trying to give them buzzwords and cool new things to try out.

Mik Kersten:
Excellent. Well, Robin, this has been amazing. Thank you so much for sharing your insights and your stories with us. Any other last things that you want to mention in terms of getting people pointed in the right direction as they map their course through the year?

Robin Yeman:
The only other thing I would leave you with is you can tell that one of the common themes in my stories is asking why or being curious. And I know that that's a common thing. But be curious, ask why. Just because you think you know it and you've known it for 20 years, think about what could be. And because of that, because technology's changed so much or we have these new

ideas, or there's all these enablers out there, you may be able to achieve things that couldn't have happened 20 or 30 years ago.

Mik Kersten:
Amazing. Well, Robin, thank you so much. And I will be following your research closely.

Robin Yeman:
Thank you. Appreciate it. All right, will talk to you soon. Bye.

Mik Kersten:
A huge thank you to Robin for joining me on this episode. For more, follow me in my journey on LinkedIn, Twitter, or using the hashtags #MikPlusOne or #ProjectToProduct. You can reach out to Robin on LinkedIn. I have a new episode every two weeks, so hit subscribe to join us again. You can also search for ‘Project to Product’ to get the book and remember, all author proceeds go to supporting women and minorities in technology. Thanks, stay safe. And until next time.