How to Improve Your DevOps Approach with Flow Engineering

Wed, Oct 12, 2022 6:57PM • 50:20

SUMMARY KEYWORDS
devops, flow, organization, dependencies, business, engineering, stream, bottleneck, transformation, value stream mapping, systems, actual, devops transformation, emilio, identify, outcome, map, capability, narrative, understand

00:00
Welcome, everyone to our webinar for this month is on how to improve your DevOps approach with flow engineering. I got my distinguished guest here, Emilio Sario. And he's going to be able to break down for you some very good practices on how we can go about looking at our organization as an opportunity for DevOps transformation. So, without further ado, go to the next slide. And we'll kick off some of the material here. So, I myself, I'm Charles Maddox, I'm Principal and founder of the i Four group, and we are a consulting firm. And we do DevOps transformations, Agile transformations, training, and the like, we've been doing that for the last 10 years here, we're based in Dallas, Texas. And yeah, we love to share more about our company with you. And then as as I mentioned, we have Emilio Sario. Emilio, when you get a brief introduction of yourself and system rescue moms,

00:55
thank you. Thank you. Nice to be here. Thank you, Charles, once again, for the invitation. While system has Manos has been helping socio technical transformation since the early 2000s, we got into agile and now we are mainly doing DevOps transformation, coaching and consulting. Thank you.

01:13
Thanks, a video. So yeah, without further ado, we'll we'll kick this off a lot of good information to share. And just wanted to kind of set the tone. And Emilio is going to give a lot of the details for you. But I don't know if many of you know, but 75% of our major DevOps initiatives that we have going on, and companies did not hit the targets in which we set out to choose and you could see here, we're not going to meet that, you know, 75% is not going to be the target this year, as well in 2022. And that's the court system of the Gartner Gartner research that's out there. Some of the the kind of the crux of what we're going to talk about today, with you. And as you can see, all these tools that are surrounding this, this statement about lack of under a lack of shared understanding, has to do with DevOps, not really being a tooling issue at all. It's really about understanding, having a shared understanding across the organization, with the many stakeholders that are involved in value flow as a whole. Alright, so again, if you're looking for getting into the weeds with a specific tool, and its API connections, and how things are hooked together, that's just one small microcosm part of what DevOps is all about. But the bigger picture is about a shared understanding of the goals and values that we're trying to achieve together, which is a key point is how to create the right narrative for change, and how we speak about the value

flow changes that we are seeking to accomplish within the organization. So that is kind of a kind of a pre precept of what Amelia was going to talk about today. But just make that hopefully, that sinks in a little bit. It's more than just the tooling change that we're trying to do. It's more than just automation, but it's about shared understanding and creating the right narrative of change that we're trying to accomplish to accelerate value delivery. Alright, so without further ado, Amelia, would you please take it away?

03:23
Thank you. Thank you, Charles. So just to keep in line with the Charles introduction. DevOps, it's not only is a huge challenge, by itself, there's a lot of things to know 1000s of alternatives of how to do automation and re practice. And I think that most of the technical engineers that are actually delivering software, they know what it's needed. They already selected some tools, they have specific things, they know that that will improve their delivery purpose, the problem we're finding on the ground is not not a lack of technical understanding on how to prevent automation on how to implement service discovery, or how to implement observability they know how to do these things. The thing is, they are not getting enough funds and resources to be able to act on that understanding they have. So our main finding and the things that we have been dealing with in DevOps transformations for the last few years, is a lack of shared understanding at the whole organization level that enables the organization to invest time and resources in what it's needed. So I think that one approach we have been following the last few years couple of years, is to try to approach these not from a best practices standpoint, was Buddha as a change management problem. When you can frame in your organization that actually transform into DevOps and uses these practices is a change process that occupies the whole organizations from the top where they probably have business problems that they need to be solved. And they cannot see the link between solving those business problems, and the actual technical practices that DevOps needs to implement in order to be successful. So the idea is to provide some mechanism, some method that you can use in your organization, to resolve that lack of shared understanding. So this has to do as

05:40
Charles previously mentioning, with this concept called a change narrative. A change narrative is a piece of those of that children the standard that we all have at all levels of the organization. And this change narrative must must make business sense. If the change narrative thing is seen by the organization as too technical, too low level, they will probably don't understand why it's needed. So the change narrative needs to be meaningful on the first side, on a business level, why do we need to invest in test automation, because we want to be able to deliver software 10 times faster, and we are doing manual testing. And that is slowing us down. So that clarity, that simplicity, shall be part of the statements of the change narrative. So a change narrative needs to be clear on opportunities, what's the opportunity, accelerate the delivery of value, what's my it agency, and that what's in it for me, for the developer, probably what's in it for him is to be able to test his software early on and shift left all the testing and be able to act when he has time to correct teams before they go into production. And that's going to improve his quality of life. For a product owner, probably he's going to met earlier, his goals are or will be able to fulfill that for at the management level. They want business outcomes, they want income, they want subscriptions, they want some business outcomes. So we need to be clear on why people need to change. That's their agency, we need to be clear on what's the common opportunity we have. And we need to understand the leverage points, we have to enable that change in the

organization. So these changes narrative should include all of this, I love this flow engineering methods, which I recently about a year ago. First, you have to know because it provides a change narrative that is appropriate for DevOps transformation. And as a side note, there is this DevOps transformation narrative that we have been working on, because it is it is transformation. A transformation, by definition is to completely change something into something else, and probably preserve some specific functions, but a transformation, a whole transformation that they were there for, they were showing damage damage. The thing with that change narrative is that is too soft. When I go to work with technical leads, DevOps engineers, solution architects, and I begin to speak to them about transformation. Nobody wants to transform, they have been 1015 20 years in the industry, they are experts, they know what they are doing. They don't need to transform to something else. So this change narrative, when we just put position that as a transformation, it lacks engagement with the technical people, because we engineers like to solve problems. And to solve problems, we do need a method, something that is clear for us, that despise based on facts. And that apply in hypothesis. validation of that hypothesis could help us to see progress. It is just another way that engineers do things. So I love I love flow engineering, because it approaches the same problem, how we transform our delivery pipeline in order to accelerate the speed of value, but with a method with a clear and simple method that

09:25

let us gain that shared destiny. That's that's I think, the most important part of this part of this change narrative that is included inside flow engineering, ESA is a layer of a basic ground theory in which the improvement needs to be done. This ground theory for flow engineering is a theory of constraints. As you may remember, the theory of constraint comes from limb thinking, and essentially like the idea is to recognize that in every flow of body In every flow of things, there's always going to be a constraint, a bottleneck. This bottleneck, the theory of constraint says that the most economic way to achieve value is to try to resolve that constraint to resolve that bottleneck. And apply in theory of constraints to a continuous transformation provides a logic narrative that is based on facts. So how are we going to transform our delivery pipelines to achieve better flow value, we are going to do it by removing them constantly and sequentially removing the the major constraints we have, that's what makes economic sense. That's something like a CIF tag, a CFO can understand, we are going to do this because we are going to achieve maximum value economic value out of transforming this list. If we get that agreement with business, we are not set on our on our way to success, because that's the biggest challenge organizations have right now are how to improve the DevOps, they need investment, they need training, they need consulting, they need some kind of solution with but they are not good at getting the resources with that with that, because there is not an economic framework to understand that that transition. So another base, in which flow engineering is in frame is in this new. And I find it quite hopeful that this new wave of popularity for value stream management really stays because at least in the software development work, we deliver value by a sequential set of steps, that is a value stream. Every developer development organization is a development value stream. And value stream management. Essentially, what it tells is that you need a series of principles to be able to manage, for example, reducing the flow time or the lead time for when you get a request and you solve that request for your customers. Reducing that list time is a principle principle of optimization and Value Stream Mapping. Anything that affects that lead time is some opportunity to improve efficiency. So when the management team and the technical team can agree on these principles for value stream matching mapping, then we are at the point where we can use flow. Why Why don't they use this because we're

going to see the maps, they're really simple, you're going to learn a master them in this webinar. They're really really simple techniques. But if implemented with the lack of proper context, without correct change narrative, without the agreements, on the business side on what makes economic sense to improve, and how this is linked for with the business outcomes, things are going to stop. So that's one of the opportunities we have this is a very simple way of doing things or gaining touch or understanding. But we need to prepare the audience to be able to do this successfully. So what's flow engineering? Well, flow engineering is this concept that was invented or promoted created by a separator they got this book when which they used to claim these four steps, I did a little modifications, because probably the context have been applying this doesn't quite fit the separator context. But as a general framework, these women are is based essentially in the work they developed. So one of the changes that I see is that

13:46
flow engineering is start with the outcomes with clearly defining what are the outcomes for the transformation we need, or the improvement with it? The thing is, where do we find those outcomes? If you go to a developer and you ask him a, what would you transform in your delivery pipeline, I'm sure that he will tell you things that will speed his rate of development, some some things that make him develop quicker, but he will probably will not be thinking on a system level. For example, most of the organizations he spent a lot of time with the acceptance criteria of the stories and there was a lot of feedbacks and QA, because that acceptance criteria is not clearly understood. So what help it is going to do to our automate their cycles when the most economic sense will be to improve the practice of acceptance criteria, and remove completely that overwork. That's the things that we need to understand in order to be able to propose a valid business case. What I have been doing mostly in a safe context, which is what I where I do most of my A work is to understand that we need to leverage on that change narrative the business needs. And what I mean with that every organization's do have some kind of operational value stream is the way they they provide value to the end customers is the way in which they get paid for the services product they're doing. And these operational value streams is supported by information systems, these systems normally do not fit the solution criteria for ideally solve this operational value stream, that's where they need changes. That's where they need they have arrows they need to adapt to a moving target, and that moving target is the best way to provide a service. So business understands this. And they understand their specific systems, not all the systems organization have the same characteristics, that there are some specific systems, that when you apply theory of constraints to the business outcome, they will be seen as bottlenecks. How do we start firm DevOps transformation, we focus ourselves on those systems that are actual bottlenecks for the operation, but that's the way we are leveraging those pieces needs to enact the transformation. So in those systems, they have development value streams, these development value streams are the group of people tools and processes they have when they need to go through the code requirements need to go through in order to be released, this development value streams do have a specific problems and their own, they have their own specific bottlenecks. This is the level in which flow engineering is applied at the level of the development districts. So in essence, my first recommendation will be before going into the flow engineering side of things, please be sure that you link and leverage the business needs by applying the prioritization of your transformation on those development value streams, that are actually solving something that that the business needs this, that that's going to make sense of the business side and become meaningful for the organization.

17:21
Oh, so the process is really simple. Usually identify the most important operational value streams in your organization, usually identify the systems and the people and then divide the development value streams. In those systems who present problems. That's, that's that's the opportunity to apply flow engineering, because flow engineering is going to give you an outcome, which is a not an actual roadmap, and how to transform that, to comply with this with to solve those businesses. So flow engineering, in a very simplistic way, is just the elaboration or four different maps that you need to do with your team and your business partners. They all need to understand what's the outcome they want to drive. That's the first mapping. After mapping the outcome, which is by itself, driven by the previous conversation on the business needs, those business needs, should be driving these outcomes that we want to achieve. After we understand the outcomes, we need to understand what's the development value stream or the flow. That's why we do value stream mapping value stream mapping is a mapping procedure I'm gonna explain later. But it's essentially a whole team exercise, when we identify where or when, or what bottlenecks are weightings. And we propose a future state with solutions to do that. This is great. But this is this doesn't take into account the actual development value stream mapping the dependencies that the value stream or the teams must will probably have from other parts of the organization that are not part of that continuous value streams. So that's why we need the dependency mapping. The dependency mapping is going to help us identify who else do we need to include in this change initiative, what we do need from them and what's the part of the transformation, either on the team or resources side, or even on the systems API and development is stuck side of things. After we have identified that we now have problems expressed as bottlenecks in the value stream. We now have dependencies and agreements that we may need to do with other parts of the organization. Then we do a capability mapping in order to solve those problems. We need to understand where we are at. We shoulda never, never tried to to implement something because a book tells you so are our best part. act extends useful, because you know, you may not be mature enough to do the kinds of practices. So, a capability mapping will give you sequential flow of improvement that is going to help your organization focus on some specific changes that our the level of maturity, one of the things that we should probably be aware of, is that the number one reason why DevOps transformation is not succeeding is because business doesn't understand why why is needed, and it doesn't get funded. The second one is we put too much emphasis on best practices, the best practices depends on context and maturity of the organization. So we should strive to provide a continuous improvement flow, not a target in a best practice, but whatever is needed. And that's what I love flow engineering, because flow engineering gives you clarity on that if you focus on that, and it's really easy to implement. It's just a few hours and some specific meetings. So, starting with the first map, maybe some of you are familiar with the Ichikawa method of finding root causes, this is something like that, we need an outcome, which probably in the case of this example, is weekly feature developer delivery, we may not be there, why because we we do not have a very automated process, because probably testing or actual deployment to some environments is not is not managed by the actual development team, but other parts of the organization, and we have queues and tickets to do. So why why are we not been able to do that weekly delivery. So these, this, why this reasoning on why we are not going to do that this is the conversation that we used to have in this specific macro, we should go without your themes, and let them freely and democratically express what they want. That's why using a tool like Miro, or some other tools that are diagramming that

22:24
allow you to do this in a collaborative manner is really essential to that after we identify a good consensus of what are the main causes, or because we are not being evil, that make us unable to enable weekly feature the leader, then we identify the context, while isn't loss, well, what are the obstacles, something like context switching maybe, or DevOps resources that in 25 projects at the same time, and there are lots of context switching, maybe we have technical depth, maybe we have a network structure that is a monolith. And most of our practices do not fit that maybe we have a lot of support. The system is not that they're stable, we are probably introducing errors to the data. And there's a lot of operations requests to fix back end data. Or we simply do not see where the things are going. Every every of every one of these obstacles should be investigated. potential solutions should be proposed as a general overview, how we should approach that flow engineering helps you with that investigation frames that initiation in a way that is easy to repeat, and can be kind of a continuous process. The second map is the value stream map, the value stream map probably you have been heard Value Stream Mapping before the value stream mapping is essentially just mapping the series of steps that your team has. And then differentiate differentiate between the actual activity time or the time that is actually spent doing some activity for example, in this case, 1.5 hours to do a sprint planning and then identifying the wait times the wait times are the times that the actual unit of value in this case, user story waits before someone else takes and get an action on that. So when you sum all the activity times and the waiting times and get our lead time or general lead time, if you divide the activity for the whole time, then you get the process efficiency rate. What that means is quite normal that before doing Value Stream Mapping development processes processes has about 50 10% efficiency, what that means that 80% of the time, there's a fix or a new feature is the spent to delivery is actually waiting time. That's the essence of value stream mapping to identify where you are losing time and to propose a future state by reducing, firstly, those wait times in order to improve efficiency, reduce the lead time without overworking without doing more work, because essentially reducing wait times is improvement for free. Probably their policies probably is there's some kind of automation we can do. But most of the wait times in reality are due to internal SLAs way of doing things, decisions that can be made in a in a work room. And that kind of things is what really helps Pakistan mapping so the outcome of the value stream mapping for you will be something like this. This is a real example of a value stream mapping in an E commerce setting where you have on the top, the teams that are part of the development value stream, the phases, name it as they are, you need to use the names that you actually are using, then we have the process times the lead times the actual weightings and the percentage of completed and acceptance work. What this means is the percentage of things, stories, requirements, whatever that go tickets that go to this flow, what percentage do they need to return. For example, if eight of the discovery items get on the next in the next phase and do not require rewarding we have completed an acceptance rate of 80%. This this top part is the actual value stream mapping exercise. What I normally would like to do, and this is kind of an improvement to the technique is to put down here to ask the team to put down here, what are the main pain points they're having in each face? tried to explain why do we have the solo sufficiency for the arm from the own subjective experience. So it's a really meaningful exercise for the team. My advice here will be to try to create

27:21

space with psychological safety. What I mean with this, developers are not very known by be open minded or freely express when they think they normally think that we are done better than them. So most of them will not care to explain something to someone who's not going to understand what they're saying. So they captured, we need to frame this value stream mapping exercise in a way that they understand that they will do will provide clarity and fact based clarity in order to improve their quality of life. What I say what I tell this, because this read cards, which I recommend that you put them put freely about what they are experienced experiencing faces, most of the things management will never see them. So we may be experiencing some pushback and friction between our team members and other areas that we do not know. Because they don't speak for that, or they think that that is not something that could be changed. So they just do not see them as improvement opportunities. So including also older themes are part of this. And framing this as a way to understand what do we need to change in order to make our life all easier, is really the the good setting to do this. To do this. I really enjoy facilitating these kinds of things. Because most of the teams I get in touch with is the first time they are doing something like this is the first time they're trying to create that shared understanding of what needs to be done. And it's always a nice experience to be there. So just just on the timing, probably the the outcome map will take you probably between 30 minutes and an hour if you have specific things. But if you're going to do the value stream mapping, as we have been telling you, you probably should receive reserve between two and four hours to have this be this meaningful way of doing things of implementing this probably you can separate these into sessions. Once you analyze the actual value stream and gather references and then you can do another two hour session to explain and go through the veins and try to get consensus on how this is affecting your policy. So the next session will be to map the dependencies and interactions. Every team in a big organization and my customers are mostly fortune 500. Companies that do have hundreds of developers, and really big delivered value streams. And there are a lot of areas and shared services teams and governance and compliance teams. And everywhere everybody's trying to max with you. And you need to ask permission for a lot of places, even if you are in Agile, because governance doesn't get up. With agile, there is still governance needed. So understanding what kind of interactions everything have with other teams, what kind of tooling they're having. And being clear which dependencies you may have to enable that change on other external teams is of essence, before even trying to provide a solution. The idea is to implement flow engineering in Agile, lean agile management. But that means that leadership should have the clear understanding that this workflow engineering is going to identify dependencies for other parts of the organization that are needed to enable change. So apply in system thinking, they should be able to provide you support to call them on the resources tab, then sit on the table, the Expand explain the business case, why the change narrative, why do we need to change what he did for us was the opportunity you have, and then get an agreement for all the other teams to help you with this in a shared program of improvement. So that's the essence of the dependency map really useful. There are some other interactions that probably you could map, sometimes you consume things, then you provide those things to other peoples that are not quite a dependency. But thinking on the customer. First, probably there are some improvement opportunities for the way you deliver software to them. So please think along the all the possible interactions you think may happen, not just dependencies, because there's some insight in mapping those dependencies. Definitely. I'm

32:14

sorry to interrupt. Just a quick question on on this, and how it might relate to the Value Stream Map. I've found in the value stream mapping activity, that some of the dependencies and some of the bottlenecks are because of the dependencies. And some of the you know, the delays are because of the dependencies. So when, when you're doing these flow engineering steps, do you have a connection that you maintain between the dependent dependency map and device map just to show that connection of time, our time is a factor. And when these dependencies get worked out,

32:50

think thank you chill temperatures, I think that probably, this will help us to provide a further detail on probably the main differences. When you do the value stream mapping of things normally do depend on some specific roles or functions in order to be able to deliver value. For example, if the if a business doesn't get you data, to do our pricing algorithm, because you don't have the categories and the data to do that, you're not going to be able to deliver that. But in other cases do depend on a more functional level on the day to day work on the DevOps, people probably enable something for you to be able to release that those are two different kinds of dependencies. In value stream mapping, we identify who in which part of the process must complete something in order to have the flow of value. In this case, the business people who needs to get to that. On the dependency map, we are clearly focusing ourselves on the day to day operations, things that actually consumed some some continuous time in coordinating logistics on day on a day to day work, because that's what in essence, who the people may need to change the directions with you may need to change the directions in order to be able to achieve flow and there are some decisions to be made. So I will frame it at two levels. We have content dependencies or actual software dependencies that should be addressed by definition by clearly defining the definition of gun for each phase of the value stream. So we we cannot start things that are going to be waiting because we have unmet dependency on the actual and the actual requirement or some other piece. That's part of the management for example in safe we do that in the PA planning. But these dependencies that we're focusing here, our day to day work, are no Meanwhile operational, constantly taking some of your time on doing that. So I think that's the main difference. I don't know is that is that a little more of Jensen? is a little confusing, but let's continue. Thank you. So anyone if you have some other questions, please. Great. So and Charles is going to find a way to get it to me because I cannot see your questions, but think. So, we have mapped the we have mapped the value stream, why identify the waiting times we have made agreements with them and how to change that to a future state. And we have shown in a fact based way, why we need to prove, then we go to the dependency map and we we identify who we need to work with, in order to enable this since then, with them included with the ones that we have just identify, then we have to find opportunities. And that's the function of the capability map. Probably one one question that may arise is, okay, what are the predefined set of best capabilities that are material DevOps organization may have? Well, there is no list of that it really depends on the context, or the maturity of the team. But someone probably could be a tech lead, a system architect, or an external DevOps consultant can come with you, and identify which capabilities are the ones that you need to deliver in order or improve. In order to solve the value stream problems, you will need help from your dependency map. Because those are other stakeholders, we need to affect change. So it's totally common, actually a good practice, to imbibe them to map the capabilities, you can you can do this in various ways, the proposals that a steel operator has employed engineering is to just average to get up point on how complete or mature we think we are, which is full of those capabilities. For example, automated testing, integration, testing, manual testing,

and then try to prioritize the ones that we have been scoring the lowest, it makes it those make some sense. If you have worked previously with maturity models, and rather ourselves, like we have in safe, you know that in order to improve your maturity, you need to have at least have the same level on the practices. So a good opportunity probably is not to find the lowest ones and try to improve them a lot. But just to the level of the other practices. So all the practices continue to evolve in the same maturity level. That way, you're going to enable some,

37:47

we're going to prevent cognitive overload, and change phatic in your teams, because if we asked for them to go to a higher level than is required, you are essentially not using theory of constraints, using theory of constraints to transform is essentially identifying the bottleneck and focus all the force on the organization and that single in that single bottleneck. So the capability map helps you to do that. It provides you a guide on how to make a roadmap and improve. And that's also access six or seven. Essentially, after we do that. After we do that capability map, then we can create the change roll up. And on the change roadmap side of things. I use have like general recommendations on how to do that. You have the practices, you should be clear on how are you going to prove this specific practices, for example, for a team that is having constant problems, because the actual acceptance criteria is not being understood on the same level by business leaders and technical people. And that really gets too complicated. On the technical side, probably getting some training on behavior driven development, and enable some kind of solution of BDD will be a priority for them, because that's the bottleneck where that's what they are wasting time. But your organization may have a different bottleneck and should probably focus themselves in different things. Just in a in a change management perspective, depending on the size of your organization. It's been known for some years, that you need at least three cycles of management cycles to be able to see if a change is going well or not. So please think about that. Our proposal, what we have normally been using with our safe customers is to include the cadence of of improvement into the program increment cadence. So every program increment, you should have a list of DevOps transforming initiatives, or things that you're going to do They're getting funded. That's why you build charter, they're standing with your business. So when you get to the planning, you are getting the time to do that. So these enablers, the idea is to commit only the short term. And then do again, at this boundary, do another flow engineering exercise, verify if your balancing has really improved, and then decide what to move on, try to not do long term planning, because that really doesn't work. But three months, eight to 12 weeks, I think is a good timeframe to implement things and try to find out if they work or not. Finally, just a ratio of this is this is the six part process flow engineering is from outcome to capability map, I do highly recommend leveraging business needs, I think that's the first computation that always will be had before initiating anything, any kind of transformation. And then I do recommend you to synchronize your change roadmap to whatever planning currents you have in your organization, because that's going to get you in that flow in the management flow in the business prioritization flow. So you don't have a separate threat of DevOps transformation that is not actually part of the actual digital operations or whatever of your business. And that's what I wanted to share to you. Thank you. Thank you for your time, I found I hope you have found these things useful. And I'm happy to answer any questions that they may be.

41:30

Thank thanks so much, Emilio. So yeah, so now the we're officially done with the webinar. And so anybody that's online, either LinkedIn live or attending the webinar live, please jot some of your questions in the chat. And we will answer those. While those questions are coming in Amelio, I got a few questions that that might be helpful to for the audience. So starting out right from the beginning, you mentioned the change narrative, who do you recommend being a part of the discussion or the sessions to help the organization generate that change narrative? How does that how does that get created, typically an organization?

42:11
Well, one thing that for sure, needs to be included in that in that, in that conversation is the technical leadership of the organization, that change narrative, the first stakeholder that should be quite conscious of the effort, even if activity are not of the current change narrative will be the IT leadership, I mean, it leadership because it should include development, leadership, and operations, leadership. Essentially, they know what they are lacking. And they will be the first interested in creating a change narrative that gets things on board. The other thing will be the main business stakeholders, the leaders of the operational value streams that are facing business challenges, I think that that they need to have a common change narrative that is agreed on those two sides of the business is really a critical success factor, the conscious construction of that change narrative, not as something that just appears, or some something that is negotiated with the actual conscious exercise of trying to find the best change narrative, you can, that's going to enable the things to get moving, because then we can align to the change relative, and we can reduce friction and doing that.

43:39
Great, yeah, thanks for that. Also, you know, you mentioned glad we got this picture here of the different maps that are created during the flow engineering process. Can you just maybe just real briefly, kind of, you know, walk through who the audience is for these typical steps and what the time commitment is corne organization to maybe develop these artifacts in their organization?

44:09
Sure, sure. I think that having tech leadership and business use on the on the business side, I think that should be the basis of the conversation. I normally do not recommend going going forward in a transformation initiative, if we don't have that agreement within business and technology that that thinks needs to change. Now, who is the actual facilitator or change agent the 50 will really arise depending on the organization, but it needs to be a technical manager, because we are going to discuss technical things and we have a technical robot to do that. So we have a we should have a single threaded owner of the transformation process the change agent. Normally when we work with customer, we ask that even though we support or the facilitation and go through the process Since we I was asked for someone who is not knowledgeable on the delivery pipeline, and understands enough of the value stream to be able to organize the things. Now, the business needs to be done with business leadership, and IT leadership as a, as a team. It may take probably one hour to understand whether things are one to two hours, but the things are going the outcome map, the outcome, man should be a lower level interaction, then you should include tech leadership, who is actually representing the business needs. And I will act as a product owner for the change initiative. But then the outcome match should begin with a whole team exercise. We include the whole team here, because they need to have the same

context, they need to have the same shared understanding. And they need to be able to try to identify the technical obstacles they're having. The value stream mapping is done with the same audience, take leadership, change agent and whole team, because that's the nitty gritty details of it. In some organizations, I've seen the functional leadership is actually impeded of going into those meeting because of transparency, risk, transparency, reason, I hope, your organization's are not that like that. But it may be needed in order to enable the openness of the conversation that's needed at the value stream mapping level. So this is the whole team as society and leadership and change agents, dependency, the dependency map that the image is right here on the face said the dependency map should be done, though as the whole team, but the capability map. Also, I think I told you by the stream, two to four hours dependency maps at most one hour is this is going to be really fast. And then the capability map that will probably take a couple of hours with the team leaders that were identified and the dependency map in that session, as part of that session, the brainstorm number of the roadmap and the solution can come up with this. But normally the change roadmap will probably take some a few meetings just to clarify and validate what's the best approach to do that. And normally, this should be done by meet program increment, to be able to have something workable for the next programming.

47:26
Great, thanks for that. Still, no questions from the audience? Have I got one more question though, and and say, you know, I'm listening in I'm here, and I'm liking what I'm hearing. I'm a engineering manager. And I'm like, hey, I need I'd like to get some training on this flow engineering concept. I want to take this back to my company, you know, what could I what could I what could you help us with on that is there any

47:57
will be to contact us, either system as a managed driver group, we are working in partnership. And in this regard, and we will help you, we will quickly assess the status you are on a flow engineering is a good fit for your company at this time, you may need to do some previous pre work in order to be able to do something like so when you're nearing job search. Flow engineering, job search is a two week process when we start from where you are, and you get a plan on how to implement that in the following six to nine months. That's essentially what we do. So Carlos, I can probably provide some other links, some information as reference or some other things that you may need to to research before doing that, but we will be happy to help you in any way we can.

48:46
Thanks. And yeah, and I know you're gonna say those things so many times. So yeah, when this webinar ends, you'll get a lot of information sent out to you on how you can get this jump started, and how your organization can take it to the next level too. So a part of understanding and learning the flow engineering process yourself is something that you can continue to facilitate within your organization even after this this Jumpstart so it is a an enabler for any internal facilitators and managers who will be able to take this forward as well. So you kind of get trained on the method and how to deliver and facilitate the method going forward with the artifacts and the maps kind of relatively created for you and a template format so so great, thanks for that Emilio. And yeah, awesome presentation. And as always, you're you're the you're the master of these topics and we'd love to have you here for these webinars. So with that said, I still don't see any other questions coming in from the audience. But we again system

as tomatoes, I for group definitely we love to help you out and answer any questions when you get this link sent out to you feel free to ping Emilio or I on these two topics of flow engineering and DevOps and I would like to load a follow up with you and just answer any questions in general. So with that said, thank you so much, Amelia.

50:09
Thank you. Thank you for the invitation. And thank you.

50:12
Thank you, y'all. And you guys. Have a great day. Take care. Bye