EPISODE 01

[INTRODUCTION]

[0:00:00.3] JM: The development of self-driving cars is one of the biggest technological changes that is underway. Across the world, thousands of engineers are working on developing self-driving cars. Although it still seems far away, self-driving cars are starting to feel like an inevitability. This is especially true if you spend much time in Downtown, San Francisco where you will see a self-driving car being tested every day.

Much of the time, that self-driving car will be operated by Cruise. Cruise is a company that is building a self-driving car service. The company has hundreds of engineers working across the stack, from computer vision algorithms to automotive hardware. Cruise's engineering requires engineers who can work with cloud tools, as well as low-latency devices. It also requires product developers and managers to lead those different teams.

The field of self-driving is very new. There's not much literature available on how to build a self-driving car. There's even less literature on how to manage a team of engineers that are building, testing and deploying software and hardware for real cars that are driving around the streets of San Francisco.

Mo Elshenawy is VP of engineering at Cruise and he joins the show to talk about the engineering that is required to develop a fully self-driving car technology, as well as how to structure the teams that will align the roles of product design and software engineering and testing and machine learning and hardware.

Full disclosure, Cruise is a sponsor of Software Engineering Daily.

[INTERVIEW]

[0:01:47.9] JM: Mo Elshenawy, welcome to Software Engineering Daily.

[0:01:50.3] ME: Thanks for having me.

[0:01:51.6] JM: You are working on self-driving car engineering. Describe your approach to managing engineers within a self-driving car company.

[0:02:01.9] ME: I think of it as leading, not managing. In a space like this where it's unknown, we would like to get the smartest, brightest people we can get and encourage them and enable them to do the best work of their life. That's the main guiding principle for us. I try to structure our orgs in a way that serves that purpose. I go with the architecture structure, then people. While people comes the last in this sentence, it's actually the first, because we try to fit people where they succeed the most based on that principle.

We make sure that our architecture matches really the org structure to avoid Conway's law, to avoid creating unnecessary layers and headaches and territories between teams that are not suitable or not feasible. Then get the right people and put them in the right place for success. This way, they have a career path and it's a win-win for them, individual growth and as well as for Cruise. That's an overall guiding principle for how we run the orgs and I'm happy to get into the details of how it's actually structured for self-driving cars.

[0:03:06.1] JM: Well, when you think about the domain of self-driving, it's a very new territory. There are companies that have been built in hardware, obviously. There are companies that have been built in software. There are companies that unify the two. You last worked in Amazon. I think of Amazon these days as one of those companies that is capable of unifying the two. Other past examples maybe include Apple. Are there things that you can borrow from other orgs that have successfully merged hardware and software, or do you feel this is very, very greenfield?

[0:03:42.9] ME: It is a greenfield. Having said that, there is a lot that I've learned from Amazon and running my own tech startups as well. Combining these two experiences together helped me the most at Cruise here. While it is a greenfield, one of the main competitive advantage for Cruise is the deep integration with GM in this case, which is deep integration with hardware. Our sensors and compute are designed from the ground up with AV software in mind. Unlike another integration model, we take that integration opportunity very seriously. We're very connected with our hardware teams.

Despite the different words between an agile, fast-changing pace software teams and more planned program-oriented hardware teams, we've managed to bridge that gap pretty nicely through program management and acceptance criteria and milestones and so on, so this way we can get the best of both worlds.

I would say that definitely, Cruise falls in this category you've mentioned where it's combining the best of both worlds, of taking hardware, manufacturing and being backed up by companies like GM and Honda was hundreds of combined years of experience in manufacturing hardware at commercial grade and giving the insights to design from the very beginning, the next generation and the generation after that, coming from AV software is a huge advantage that we're taking to the fullest.

[0:05:06.5] JM: Tell me more about the communication boundaries and the interfaces between these different teams, because you don't want to overwhelm the different teams by having too many requirements, too much shared expectation around what no hardware team should know, versus what should the hardware team know from the computer vision team, for example. You want to have those boundaries well-partitioned, well-defined. How do you set those boundaries? How do you find the right communication structures between these very disparate teams?

[0:05:47.5] ME: Yeah. It's a great question. Between even software teams, you know that these problems exist if you don't define clear charters and visions for teams. One of the first principles as you do your old charts and your old design is to really think about the team charters, like what's the mission of this team? From there, the vision and the charter of a team, from there you can compound that into a very clear charter, or boundaries of what this team is responsible for.

The boundaries define the interface between this team and every other dependent and interacting – how they interact with other teams. From there, between software and hardware, specifically transfer your question, for example, you're right, there is two very different words, as far as the way things run in agile methodologies in general are not very well-adapted on the hardware field, because you need to plan ahead, sometimes years ahead of what exactly needs to be done components-wise.

We bridge that gap through again, milestones and acceptance criterias, which acts more of handshaking points between the software teams and the hardware teams. We have mechanisms in general that makes sure that we stay aligned as a company as we grow. Those range from the opening up all the backlogs for all teams, for example, so that every team can know not only what this team is working on, but what's coming their way. They can attend their backlog grooming sessions and add things, so that we stay agile.

We have execution, weekly execution reports. In Amazon wars, that would map to the WBRs, the weekly business reviews they have. We have a mechanism that we're actually running this week called PTRs, product and technical reviews, where the partners across the company present what they've done in the past six weeks, what they're about to do.

This is a great opportunity for us to build the culture we're set to do, which is a peer-driven culture that is fully aligned. It's an opportunity for leaders to come in and first reflect on what they've learned. To pause, it's a strategic pause. Get out of the tactical hamster wheel and celebrate wins, hold each other accountable, learn from our mistakes and our misses through the past and then align on what's coming. When you have these mechanisms along with a good, well-defined charters for teams and boundaries and organizations, you minimize the chances of misses, or conflict.

[0:08:19.3] JM: There's a picture out there that compares the different org charts of different companies. It's a comic. I can't remember the source of it, but the way that it describes Amazon's management structure is very firmly hierarchical, in the sense that if you look up the management chain from any point at the bottom, there's always going to be one direct manager and all the way up to Bezos at the top, or maybe Bezos and Jeff Wilke. It's a straight path.

At another company like Facebook, you might have less of a straight path. It's a more flat organization, where information sharing is more widely propagated and it's like a choose-your-own-adventure, what information do you want to learn, dive into the wikis, find whatever you want, find other people in the org, explore.

At Facebook, there's a less well-defined idea of what are we trying to do here? We're trying to build social networking software. That's a bit different than something like self-driving, we know

what we're doing, we know what we're trying to accomplish. We're trying to get self-driving cars going. How does the well-defined mission of the product and what you are trying to accomplish, does that make it such that you want a more well-defined, rigidly structured hierarchical system because you have the mission so well-defined?

[0:09:58.1] ME: That's a good question. I would say the two words you've described, we have a hybrid between these two, which in my opinion gives us the best of both worlds given that advantage you mentioned. You're spot on. That alignment is not easy found otherwise. Even at Amazon, by design, you have sometimes tens of conflicting goals at the SVP levels. Because the company is designed to cannibalize, sometimes even between teams and may the best opportunity wins. They can afford that bet.

With misalignments, a lot of culture issues emerge and it becomes much harder to align teams and accordingly, they had to go with the rigid structure, etc., so that's understandable. Facebook have the luxury to explore and their mission as you mentioned is pretty there is room for divergent from there. I think we have the best of both worlds. The way we've structure things here is that we are – we have this set of org guidelines, where we intentionally keep it limited number of layers. We have only four levels, for example here. We have only four titles.

The philosophy behind this is to make us focus more on to the work itself, not the non-value-added promo between one level and the other continuously. It also makes it very clear as you have less levels, the guidelines and the boundaries between these levels becomes very clear. You can ask any person what's the difference between a staff engineer and a principal engineer and they will tell you.

That clarity and role clarity provides additional guidance of what's expected of you. Keeping the org structure itself shallow, no more than four levels, makes communication flows both ways much easier and removes unnecessary layers of hierarchy as well. Beside all that, the communication flow you've described in a company like Facebook, we've put mechanisms here to facilitate and govern that.

The PTR as I described for example, this – a situation where we invite everyone, not only from this team, but from other teams to come in and participate and bring ideas to the table, push

back on certain direction that this team is coming for, ask for help if needed, so the communication flows in all sides.

We also have this very important, I call it the glue structure beside all these old structures. I can go through my org structure in details, but there is two functions here; TPMs, which is you're probably familiar with, that's Amazon have, and PMTs, which is the equivalent of a product management in a company like Google, which is very technical product management.

Those are the matrix glue that works between these teams, so that they form these spot and form these cross-function virtual themes in many cases to tackle problems that spans multiple team boundaries. Because you don't want to get lost into – tied into too much into your organization boundaries and accordingly have gaps where it would drop because it doesn't necessarily fall right into one team or the other. At the same time, you want to encourage that cross-collaboration from career growth perspective. It's helpful in so many fronts.

Our PMTs and TPMs are tasked with looking across the boundaries, along with principal engineers and staff engineers of course to form these cross-virtual teams. I really think we have the best of both worlds in this situation. We're lucky again, because there is no conflict about the mission here. There's no conflicting goals. Every single person in this company you can ask is like, know exactly what we want to do and where we want to be.

[0:13:31.2] JM: Let's make this more tangible for the listeners. Could you pick two teams that are closely related to one another and describe how those teams slot into the management hierarchy? Just talk about how those teams are structured, how they share information with each other. Does an example come to mind?

[0:13:53.2] ME: Yeah, plenty. Let's take on the AV side, on the autonomous vehicle side. For example, we have two big teams think of one as the ML AI team and that's where all the perception stacks lies, along with some of the needed infrastructure teams, like machine learning platform, ground truth and automation.

That particular team and data science as well falls in this team, works very closely with what we call the AV Robotics team. The AV Robotics teams is where planning and control happen. It's

under different leaders and the different structure, but this is where planning and controls and the actual – when you actuate the vehicle controls happens, right? After the perception, prediction and predicting the path of the NPCs of the non-player characters and the scenes into where they're going, you want to actuate the car to actually take it safe and smooth pass to the planned route.

These two teams for example, interact all the time. They're very closely together, because you don't want to – they're down the stack from each other, so you don't want for example, running into the capabilities, running too much with the capabilities of the perception team if it's not going to be consumed on streams, the planning and control teams might not be as effective.

Accordingly, our AV TPMS in this case who are the cross-function TPMs that I've mentioned, are driven by – we divided the product ownership by maneuvers, for example in this case. Not by the stack, but by maneuvers. Someone would be focused more on how the car is behaving in a certain maneuver, like left turns, unprotected left turns, for example.

[0:15:33.0] JM: You have a left turn TPM.

[0:15:34.6] ME: Stop signs, that kind of stuff.

[0:15:36.5] JM: A stop sign TPM.

[0:15:37.9] ME: Yeah. This way you can, because that maneuver by itself have a cross-functioning by nature in it, so there is always someone, or multiple people actually, not someone, looking at across the board, end-to-end across the spectrum how these teams should interact, what are the most effective projects to work on and so on.

Same happens between even unrelated teams, like data teams in infrastructure and MLP. There is very close relation, even though the machine learning platform as you know, like most of data scientists, or advanced machine learning applied scientists spend sometimes up to 70% or 80% of their time just to get the right data and cleans it and so on, right? Clear coordination between the two as far as just what's in the data lake and what quality data that we have in the data lake, also influences the work of MLP to pick it up from there and have – how to build the layer, so

where the machine learning platform is an added layer on top of a data lake platform, for example.

There is a lot of coordination between these two teams again, through TPMs and through the leaders of these teams itself. Our leveling guides, in this case, specify these concerns or responsibilities on the senior manager level and on the staff and principal engineer levels that it puts this as an opportunity for staff engineers to grow into principal engineer and so on, to work across cutting initiatives like such.

[0:17:06.6] JM: The TPM is the glue between these two kinds of orgs, between the data and the infrastructure, between the perception and the perception team and the –

[0:17:18.8] ME: Robotics.

[0:17:19.4] JM: - hardware team.

[0:17:20.1] ME: Robotics teams. The robotics team is still in software, planning on controls in this case.

[0:17:24.0] JM: Oh, planning. Okay.

[0:17:25.3] ME: TPM is part of the glue, right? As I said, there's TPMs, there's PMTs, which are product management technical, which are responsible for more cross-function, like writing product strategies. There is also the – as I said, the leveling guide for managers and for ICs, encourage you to work and look cross-function as you grow throughout your career. Intrinsically, there's no conflict. There's an intrinsic motivation for a person, whether they are manager or an IC. As they grow, it’s to collaborate, to look cross-function, to work on meaningful and more impactful initiatives and not be internal-focused.

[0:18:02.5] JM: That's quite interesting. Basically if I join Cruise, my baseline expectation from my managers, from the company is to solve a specific problem that I was hired for. If I want to level up in my career, it's almost like the vector for leveling up is how much I collaborate and how much I reach across the organization? Is that what you're saying?

[0:18:32.0] ME: That's one of many dimensions, right? I've published this in the blog. We look at multiple factors. There is the scope and impact is one of them for sure. That grows by a collaboration, sometimes it might not need to. In some cases, sometimes going deep and technically vertical might be enough. There's also the technical complexity is another dimension. The performance and execution on your own level is another dimension. The personal leadership or leadership in general is another dimension.

We look at all four dimensions. It's not just one vector as you mentioned, it is one of the vectors. Cross-function collaboration when needed, if that was the right answer for a bigger scope and bigger impact for Cruise and for your team as a whole, than great. If your team needs certain design components within your team boundaries that more impactful, in this case for Cruise, then that becomes the right path.

[0:19:24.4] JM: Tell me more about how the incentives to move up within an organization drive your philosophy around management, because I've read your article on the Cruise blog, which we'll put in the show notes. You seem to have a pretty strong belief that knowing how to level-up, how to move up and through an organization is really important for getting the incentives right, for getting the workforce properly motivated. Am I understanding correctly?

[0:19:59.0] ME: Partially, I guess. I think my main intention was before moving up really, when we go to performance calibration, right? I want to be fair to the engineers on my team. I want to make sure that everyone is rewarded appropriately for the great work they do here. In order for me to give you a balanced and fair performance review that helps you more in your career and helps you reach your maximum potential, I first owe you a very clear definition of what's expected of you.

I think when you define very clearly for every IC what's expected of them, then you remove a lot of the confusion and the conflict around performance reviews. I really want to focus on that. Most of people join us here not to advanced in their career just by – just fake titles, because then we can have had 16 titles and we can make promotions every six months. I think that's a non-value added to the person and to the company.

I think the real value add is what are the gains you're going to gain here at Cruise in your own personal career? How much are you going to learn? How much better engineer you're going to be? That's what we're focused on; defining that, defining that criteria and keeping the bands pretty wide to remove the bureaucracy, to remove the unneeded, unnecessarily promotions every six months or change titles, to remove people being fixated too much on getting to the next level, because the more you have them, the more that's going to occupy more of – more and more of your mind. I want people to focus on the work and the great matching we have. We have one of the best opportunities in our lifetime to participate in something as big as this. That was what's behind the philosophy is really provide that clarity, so that we become more enabled to help our engineers be successful.

[0:21:43.4] JM: How much time do you spend thinking about proper team structure and proper org layout, versus thinking about deeper, technical problems throughout the stack? I mean, you could spend your day in so many different directions. You could spend your day entirely looking at wikis and pull requests and understanding how things are going at a deeply technical level, or you could spend the entire day looking at whatever 800 person org chart, or I'm sure it's growing by the day.

[0:22:22.2] ME: That's right.

[0:22:23.0] JM: You're looking down at.

[0:22:24.8] ME: Yeah, that's awesome question. It resonates very much. I try to balance between three pillars. Behind anything great thing in my opinion, it's people, process and technology. I try to balance my day and my time in general across these three pillars, right? People in this case is beyond just org charts. At the end of the day, I am responsible for everyone in my org and I try to be genuine about that.

In my one-on-ones for example, my goal is to make every leader who reports to me become much better leader than myself. I can enable that by giving them all the experiences I have, so they have this advantage of starting where I haven't. You spend your time between organization charts, between team charters and missions, between what new opportunities have we've

learned that might inform a better team structure in certain areas; one-on-ones, alignments, all the stuff falls under the people category.

Processes in my opinion, this is a loaded term, because many people would interpret that as bureaucracy, or as unneeded steps that makes things hard. I think of it it's exactly the opposite. When I say processes, I think of mechanisms that makes life easier for everyone, for engineers, to govern information flows, to put things in place where it's very clear what's the expectation is. As we grow in a hyper-scale growth, communication is key, so how do we make sure that teams are transparent? Because transparency drive efficiency. How the team communicates with each other? How do we make sure that teams are aligned, so that everyone is delivering what's expected of them, holding each other accountable and so on?

Thinking of all the mechanisms we put through in a way that would remove bureaucracy and make us all better collectively. For example, we value and encourage blameless post-mortems. It's a very good way for us to learn from mistakes in a blameless way, so that we build and become better engineers as a whole and a better company. I try to focus between that as well, people and process.

Then technology, we also have a process for that. We have design reviews, for example that goes in. There's wikis for upcoming design reviews, which teams are putting their desired reviews and there is guidelines that you need to submit it before the meeting. There's a meeting where everyone is invited to the design review, every relevant person according to this team can show in and provide insights, provide feedback.

Sometimes great things come out from these design reviews, or sometimes people would comment on the lock itself before it even goes to the design review meetings. I spend my time as well looking at that from a technology perspective and putting guidelines there.

[0:25:06.6] JM: Give me a description for how an engineering team can productively create a design review for other people within the company to see? Because this seems like a very important area of cross-team communication. What goes into an effective design review?

[0:25:28.8] ME: We've published a very clear guidelines that goes into details of what each section is supposed to be and what it's covered, but I'll try to go give an overview of that. The main thing is we want to start with really what problem are we trying to solve. I can't put enough emphasis on this, because most of the time this is the difference between efficiency and effectiveness, right? We want to make sure that our team are effective. They're running in the right direction, not only running as fast as they can, right?

The effectiveness here is really getting deep into the problem statement, what problem are you trying to solve and more importantly, why, and then why now as well, because there's no shortages of problems, or technical problems, or opportunities where you can make things better. Prioritizing that and making sure that the entire teams are aligned and your dependent downstream and upstream teams are also aligned on that this is the right time, this is the highest priority to tackle at this point being backed up by data and KPIs, which we do have and publish and work on every day is key.

That part of the design document is pretty strong and pretty useful and sometimes it's linked in to more detailed product requirement docs and so on written by the PMTs, but this is an essential section. Then goes into the technical design and the technical design of the systems, whether it's logical, or physical deployments. We go into sometimes in the non-functional requirements as well, which is very important.

Most people focus on the functional requirements exactly, like I want this component to do X, Y, Z. As important, there is the non-functional requirements, like what's the expectation of latency for this component, for example? What's the volume and scalability requirement to support this component? What's the security requirement?

Security is an integral part from everything we do. It starts at the design time. You're always sitting in the design reviews and we're taking that in consideration by design and deployment and development, of course. It covers that section pretty good and then it goes into what's in scope and sometimes it touches on next phases and so on.

[0:27:42.9] JM: When I think through the lifecycle of the Cruise product, so I see Cruise cars throughout the city. I live in San Francisco. I've been seeing them ever since I moved here. I

know that there is a life cycle between the car getting updated, the cars driving around, they're collecting data. You're using that data to feed back into the system and improve both the software and the hardware. Can you sketch for me how the life cycle of the car testing and continuous improvement cycle, how does that feed back into the company, in the internal product development?

[0:28:36.0] ME: One of our first principles here at Cruise is safety. Everything I'm about to say is basically guided by that very simple principle is safety first. How can we make sure that before we release any code and before we focus on iteration and fast iteration, we make sure that we don't violate that very important principles, which is safety first?

We have a very rigorous release process here, a very rigorous closed-loop process, where we take testing and simulation and account and we take it test first, test-driven mentality, where as we get things from the road, or as we work on the next feature, we have clear guidelines of how are your PRs being tested, from a unit test perspective, from an integration test perspective, from a simulation perspective to run all the different KPIs that we know should improve, for example when this PR is pushed?

Then more importantly, how is your PR affect the entire crew stack? How can we run the simulation for everything integrated together and make sure that there is no regression? Only when we pass these very clear gates that I can get into in details, then we can go into a very surgical deployment of road tests and examine on the road how is this code behaving before wide deployment on a fleet, right? Then go into detailed drive analysis as we deploy this data into our cloud stack and look into very detailed metrics and interaction between our different layers of the stack, then it gets released to the rest of the fleet.

One of the things that I'm really proud of here is everything I've described is you can optimize without compromising safety. The best thing you can do in a known-unknown domain, like what we have is to accelerate the rate of iterations and the rate of experimentations, which we've done that through great tools and platforms, things as I mentioned the data lake, the machine learning platforms and the simulation platforms and so on.

We have these things in place, so that we can accelerate the cycle I told you about from road to code and then from code to road, in a way that seems pretty agile and pretty continuous. We do have the continuous integration, continuous deployment cycle, except it's much, much more interesting than as you've mentioned, the software as a service company, for example.

[0:31:00.6] JM: Well, tell me more about that. Because I certainly cannot go and buy – go to CircleCI and say, “Hey, can I get the self-driving car product, please?”

[0:31:10.6] ME: No, you can’t.

[0:31:12.5] JM: You just have to build a ton of internal tools for this, right?

[0:31:14.9] ME: That's correct. This is where the balance in organization structure is, right? This is where some companies philosophies, for example, can afford a full vertical integration and having every team do everything on their own, but then you compromise duplication, versus creating dedicated team, for example for engineering productivity, or for a machine learning platform, and have a very clear vision and charter for what is our machine learning platform going to do?

What benefits, or value-add it’s going to do? What is our circle like? What are our CICD tools, the equivalent of circle in our case is going to do? What's their charter? What value add on Circle, or Buildkite, or whatever alternatives out there we will add on top of this, so that for the AV engineer who is using this on daily basis, it's completely seamless? They just want to check their code. They want to see and know that everything I've mentioned are being done in the background and they're safe in deploying this to the course.

[0:32:16.4] JM: Now you've worked at Amazon. You know at Amazon, they build everything. They will be like, “No. Skype is not good enough. We're building a VoIP software. Slack is not good enough, we're going to build something.” These tools range in quality.

[0:32:33.0] ME: That’s right.

[0:32:33.8] JM: Internally. I've seen some strange internal software. I'm like, "Why don't you just buy it? Do you really have to build this?" They just build everything.

[0:32:44.0] ME: That's right.

[0:32:44.6] JM: It's not exaggeration. To what extent do you – I mean, it sounds like to some extent, you have to borrow that, because your domain is very unique, your domain is very new. There must be some stuff you can take off the shelf, like Kafka, or Tensorflow.

[0:33:01.4] ME: It would be irresponsible for us to rebuild those. It would be really irresponsible for us. We rely on a lot of open source stacks out there. Our pass layer is built on Kubernetes, right? It makes no sense for us to rebuild Kubernetes. It would insane, right? It makes sense for us to build a layer on top of that that would do the governance in the way we're running things at Cruise. To segment our projects and namespace and integrate with our vault and security in a certain way to make it easier for engineers.

We focus on their value added on top of what's there. I don't want to build things just for the sake of building things. Again, that's not what we're all here for. We're all here to make a real difference to the world. The real difference toward is going to happen not through building another Kafka. Again, that would be irresponsible.

As a matter of fact, part of my technical strategy is to when appropriate, if we can use managed cloud services even, not just cloud services, managed cloud services. If we can go serverless, absolutely go serverless, go manage cloud service, because the cost was seemingly higher, is much higher to actually take our valuable engineering resources the most valuable thing we have and put them into an already solved problem.

We want to direct people into more the unsolved problems. We tend to focus towards that. If there is technology out there that passes of course of our security guidelines and meets our needs and we can just build a thin layer around it to help us encapsulate it and isolate it from dependency on another third party, which we take in consideration on our design principles, then definitely that's the direction we're going. We only build things if there is no alternative for it.

Taking in consideration, again things like our scalability requirements here is very different, because we did with a large volume of data in certain cases, or security requirements is very different and so on. We consider all these factors very, very carefully.

[0:34:56.6] JM: Are there any specific build versus buy decisions that come to mind, like things where you've just been like, “I don't know what the answer is here,” and you end up building, or you end up buying and then you end up saying, “Oh, that was actually a mistake. We got to go back. We got to rebuild this thing.” Any specific technical examples come to mind?

[0:35:15.8] ME: I don't know how much liberty we have there with our providers. I'll tell you that we're using for example, both AWS and GCP right now –

[0:35:24.0] JM: Cool.

[0:35:24.5] ME: As cloud providers. We're using a lot of the services on both sides.

[0:35:28.9] JM: That's because there's interesting services. It's not necessarily for a failover case, or both?

[0:35:35.5] ME: Okay, if you're asking about the stack running on the car, all the critical safety workload that's running on the car – there's a difference between on-car software, versus what are you going to do with this massive sea of data afterwards? How are you going to analyze your data? How are you going to drive in size? How are you going to build the machine learning platform, where you can train your models multiple times a week and not wait for month between evolving with your mobile structure and so on?

For these offload things, where running your, as I say, a platform as-a-service in a scalable containerized matter where you can scale horizontally almost indefinitely, or relying on NoSQL database as well, you don't have to worry about scaler all. For these things, using the cloud makes more sense, for things like the rideshare service itself, the mobile app that our employees here use on a day basis to take the cars like any rideshare service. For these things, it makes more sense for us to move towards the cloud, at least for the time being.

[0:36:34.4] JM: I think one of the somewhat mistake and beliefs that people have about multi-cloud is that the best reason to go multi-cloud is that you can have your data both in S3 and in Google Cloud storage. I mean, that's useful, right? It's useful in case of a cloud, entire cloud fail, like AWS literally fails.

To my mind, the real upside is being able to use BigQuery, or being able to use managed Tensorflow on Google Cloud, in addition to using the proven AWS stuff for running your business, like using S3 and using IM policies. What's the advantage of being multi-cloud for you?

[0:37:18.4] ME: Our preferred cloud provider is GCP at this point.

[0:37:22.4] JM: Really?

[0:37:23.0] ME: Yeah. The advantage as you said, I don't want to overburden the teams also from being multi-cloud, but at the same time you don't want to tie your fate into one provider. There is a difference between existing in all places at the same time, versus designing your software in a way that is generic and portable, so that you can migrate if needed.

The availability SLAs for example, provided by GCP or by AWS is enough to inform your decisions there. If that meets your non-functional requirement for certain cases, as far as the durability and availability of data then okay, that makes sense maybe to keep it in one place, maybe in one zone, maybe in multi-region zone. Maybe in some cases based on the non-functional requirement you need to keep it in both cloud providers that depends on the nature of the component itself.

[0:38:13.1] JM: Can you say more about why GCP became your preferred provider?

[0:38:17.8] ME: Well, it was not an easy decision. It was not off-the-cuff decision for sure. We have done a bottoms-up analysis of all the features on both providers and specifically the things we need the most, like for example, Kubernetes support at the time was much more advanced, along of course as like, the cost of running things on both clouds.

I think our decision was basically based on our engineering needs more than anything else; our engineering needs on our time to market and the reliability of these features on both clouds, along with cost and reliability and so on.

[0:38:56.0] JM: Very interesting. Let's come back to the question of platform engineering, or engineering productivity. How does the engineering productivity team find out about what issues in the organization they should be solving for and building general solutions for the rest of the engineering workforce?

[0:39:21.4] ME: Yeah. Collaboration is key in this area. We have a great leader in this space who's very, very passionate about what he does and he instilled that in his team as well. Like for example, they send out a survey even for customer satisfaction, not only about what's next, but how do you feel about what's there? This is continuously trying to seek in feedback and anecdotal feedback.

Beside the channels, the official channels that I mentioned, which are the PMTs and the TPMs helping groom the product backlog in the first place, beside the mechanisms where the engineering productivity come in and present to the entire company, “Here is what we've been working on. Here is our hits and misses and here's what we are about to work on.” For everyone in the company to raise their hand and say, “How about this? Or why working in this instead?” These are all guided mechanisms, right? We send execution reports for what are they doing on a weekly basis to show progress and so on?

Beside all that, the key thing here is real connection and real passion between people, which is the case here. A lot of the people in the engineering productivity are really passionate about what they do, because they know the faster iterations we can provide our AV engineers and the more pain points they can remove, the faster we're going to reach our mission. They are actively seeking out feedback all the time. They're meeting with the key AV engineers and staff engineers, or engineering managers and these teams to find out about pain points and find about what's next.

There is also a cross-cutting initiatives that we’re running across the company related to that closed loop that I've mentioned, like the test-driven mentality that we're trying to adopt here and

how all this relates together between the release and the engineer productivity tools between the continuous integration and continuous deployment, which falls into this area that drives a lot of the work.

[0:41:09.7] JM: Can you take me inside a Cruise car today? Explain to me, I get into the car, we put in a destination. What happens?

[0:41:21.7] ME: The sequence might be a little bit of there. We put the destination before even getting through the car, right? That's an awesome advantage that we have, right? Because our dispatch system have – we have no humans in the cars. As you can control your supply and demand and you can even control it ahead of the curve, ahead of the supply and demand curve, which is it's going to be very interesting. It helps us achieve even more and more of the mission of the traffic congestions and working around the traffic flow.

There is a lot of opportunities there that I don't want to get into in details. Once your route is decided, the safest route to take you from point A to point B on a high-level, which is the car, the brain, the computers and the car, which is it's actually dual computer systems built from the ground up for this mission, works in a nutshell on a high-level where you're like – first, it perceives the world, the surroundings, the world around it, to understand what objects are around. We have a multi-sensor suite that's built for the car, again from the ground up to do these things.

You perceive what are these objects around you are about to do. You track it and you use the tracking to predict the object's behavior. Based on that, you come up with your plan, with your path and then you start actuating the car. Then the car takes you from point A to point B, which is what our employees here use. They use their ride share app on a daily basis to take rides on the car and go around the city.

[0:42:46.1] JM: That's a perk.

[0:42:47.0] ME: Yeah, absolutely.

[0:42:47.9] JM: You didn't even get free prime when you were at Amazon.

[0:42:51.5] ME: I think they gave everyone $50 or something.

[0:42:54.3] JM: Maybe at the VP level. I certainly did not get that. AWS credits. Something. They're very parsimonious.

[0:43:04.6] ME: The joy of sitting in this car and feeling the impact, seeing your code, pushing code and going through that safe cycle and then getting downstairs and getting in the car and seeing it happen is one of the best joys and best perks you can ever encounter, right?

Because most of us get into software engineering in general, because you like that instant gratification. That becomes much harder when you think of self-driving cars, it's not like you're building a web server and you're just going to spin it on your desktop and look at it. We have a very close system that gets you close to that in a safe way, which is amazing. It's great to see your impact on the right quality right away.

[0:43:44.9] JM: Well, when you're getting into a self-driving car, you're getting into software engineering in a whole new way, in quite literal sense.

[0:43:51.5] ME: Absolutely. Absolutely.

[0:43:53.5] JM: I mean, even in San Francisco, the most optimistic software engineering place on the planet probably, some people have lost some faith in the world of – I mean, some people think self-driving cars can't actually happen. I don't know. There seems to be this fluctuation in how much faith people have, whether this thing is actually feasible. Do you ever wake up and you're just like, "I don't know. I'm having doubts today." Or do you have a persistent faith that this is going to be solved. We are going to do this?

[0:44:33.0] ME: Yeah. Not only I have a persistent faith, I have a persistent passion. It's more of it's the opposite. It's actually the driver towards getting me to work every day is because you know you can make a difference and you can make an impact. It's understandable that many people would be skeptical, right? I think people were skeptical, even when we had elevators. There's at one point of time, there's an operator sitting there –

[0:44:54.3] JM: The operator. It's true.

[0:44:55.7] ME: Yeah, clicking the button, right? Cellphone, same thing. It took years for that to come out to market. It's not unnatural for people to be skeptical. It's our job, our responsibility, our duty is to make sure that we are deploying a safe fleet that is actually safer than human driving. When we get there, the reward would be incredible. I think, most people here are driven by that passion, not by the fear of not achieving it, but actually the passion of achieving it.

[0:45:25.0] JM: Yeah. Okay, so last question. What are the biggest bottlenecks to self-driving cars, being something that consumers use on a regular basis?

[0:45:36.9] ME: I think that ties back to the previous question, which is our passion is to put a safe self-driving fleet on the road. In order to get there, we need to pass our own bar for safety. Of course, we're working very closely with regulators and we have great teams on marketing and government affairs and all the stuff. As engineering, everyone here is know that this is a safety first mission. We want to make sure that we take a very deliberate engineering approach towards engineering a car that is safer than humans.

I don't think of it as obstacles, as much as again, achieving that greatest technology challenge for our generation and doing it right and doing it in a way. Unique to Cruise strategy here for example, is that we're starting from the hardest places to drive in in San Francisco. We don't want to paint ourselves into a corner architecturally.

We're thinking at scale level from day one, that's why we have the integration with GM and Honda. We're manufacturing our cars in plants that can produce hundreds of thousands of cars. We're taking operations, scale, safety in mind from day one, so this way you can build things architecturally speaking in a way that is not just for a demo sake, or launch in a suburbs, in a city, but then have a very hard time really getting to a commercially viable product that will change the world. We're after a true change to the world, not just launching a demo service somewhere.

[0:47:14.0] JM: Mo, thanks for coming on the show. It's been really fun talking.

[0:47:15.7] ME: Thanks for having me. Pleasure.

[END OF INTERVIEW]

[0:47:26.8] JM: You just heard from Mo Elshenawy, Vice President of Engineering at Cruise. Cruise is a San Francisco-based company building a fully electric self-driving car service. Building self-driving cars is complex, involving problems up and down the stack from hardware to software, from navigation to computer vision. We're at the beginning of the self-driving car industry and Cruise is a leading company in the space.

Join the team at Cruise by going to getcruise.com/careers. Cruise is a place where you can build on your existing skills while developing new skills and experiences that are pioneering the future of industry. There are opportunities for back-end engineers, front-end developers, machine learning programmers, many more positions.

At Cruise, you will be surrounded by talented, driven engineers like Mo, all while helping to make cities safer and cleaner. Apply to work at Cruise by going to getcruise.com/careers. Thank you to Cruise for being a sponsor of Software Engineering Daily.

[END]