EPISODE 663

[INTRODUCTION]

[0:00:00.3] JM: Self-driving transportation will be widely deployed at some point in the future. How far off is that future? There are widely varying estimations. Maybe you will summon a self-driving Uber in New York within 5 years, or maybe it will take 20 years to work out all of the challenges in legal and engineering.

Between now and that self-driving future, there will be a long span of time where cars are semi-autonomous. Maybe your car is allowed to drive itself in certain areas of the city, maybe your car can theoretically drive itself in 99% of conditions, but the law requires you to be behind the wheel until the algorithms get just a little bit better.

While we wait for self-driving to be widely deployed to consumers, a lot could change in the market. We know about Uber, Lyft, Waymo, Tesla and Cruise, which is owned by GM. What about the classic car companies like Ford, Mercedes-Benzes and Volkswagen? These companies are great at making cars and they've been doing so for 50 years, a 100 years. They've hired teams of engineers to work on self-driving. There's also a large volume of companies that are springing up that can offer software and other technology solutions to these big old car companies.

Self-driving functionality is not the only piece of software that you need to compete as a transportation company. You also need to build a marketplace for your autonomous vehicles, because in the future far fewer people will want to own a car. Consumers will want to use transportation as a service.

RideOS is a company that is building fleet management and navigation software. If you run a company that is building autonomous cars, you need to solve the problem of making an autonomous, safe robot that can drive you around. Building an autonomous car is really hard, but to go to market as a next-generation transportation company, you also need fleet management software, so you can deploy your cars in an Uber-like transportation system. You also need navigation software, so that your cars know how to drive around.

RideOS lets a car company like Ford focus on building cars by providing a set of SDKs and cloud services for managing and navigating fleets of cars. Rohan Paranjpe joins today's show to talk about the world of self-driving cars and what the world will look like as we get towards the self-driving car.

Rohan worked at Tesla and Uber before joining rideOS, so he has a well-informed perspective on a few directions the self-driving car market might go in. It was great to talk to him and I think you'll enjoy this episode as well.

[SPONSOR MESSAGE]

[0:03:00.8] JM: DigitalOcean is a reliable, easy-to-use cloud provider. I've used DigitalOcean for years, whenever I want to get an application off the ground quickly. I've always loved the focus on user experience, the great documentation and the simple user interface. More and more, people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU-optimized droplets perfect for highly active frontend servers, or CICD workloads.

Running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily. As a bonus to our listeners, you will get a $100 in credit to use over 60 days. That's a lot of money to experiment with.

You can make a $100 go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure and that includes load balancers, object storage, DigitalOcean spaces is a great new product that provides object storage, and of course computation. Get your free $100 credit at do.co/sedaily.

Thanks to DigitalOcean for being a sponsor. The co-founder of DigitalOcean Moisey Uretsky was one of the first people I interviewed and his interview was really inspirational for me, so I've always thought of DigitalOcean as a pretty inspirational company. Thank you, DigitalOcean.

[INTERVIEW]

[0:05:07.7] JM: Rohan Paranjpe, you are an engineer at rideOS. Welcome to Software Engineering Daily.

[0:05:12.0] RP: Thanks for having me.

[0:05:12.9] JM: You've worked at Tesla and Uber and now rideOS. rideOS is a company building some technology in the automotive space. Tell me about your experience building car technology and tell me about some uncommon beliefs that you've developed about the future of cars.

[0:05:33.6] RP: Sure. When I was at Tesla, I worked in a team called Autopilot Maps. We were basically taking GPS data from the cars and building these things called splines that allowed the autopilot vehicles to actually drive on them when vision wasn't available, or when conditions were bad.

I worked a little bit on the firmware, but I also worked a lot on the backend and actually doing the data collect working with how to integrate GPS data and probe data with base map data from providers like OpenStreetMap. I've had some experience, especially on the backend services for providing mapping and routing for vehicles. I worked on the routing engine at Uber that powered UberPool, UberEats and a bunch of different services that relied on accurate ETAs and routing decisions.

One of the big things I realized as I was working in both these companies is that a lot of the mapping tech stack and services are being built from the ground up every single time. I realized that a lot of this could be a separate horizontal layer and that if we're going to deploy autonomous vehicles quickly, safely and more efficiently, that would be in our best interest to have one place where new companies can start up and once they want to start deploying fleet technologies, they can talk to one place or one company to get that going and get the ball rolling.

One of the big things that I've realized also in addition to this is that I think the ride-sharing model is going to take off in a pretty spectacular form. I think it already has with regular cars, but I think a lot of us still are assuming that we will own self-driving vehicles in the same way that we own our normal cars at home. I think this is going to be not – or I don't think this isn't necessarily going to be the case.

For example, we have a lot of different requirements for our own personal vehicles, right? For example, we expect our vehicle to take us not just within one city. For example, I expect my vehicle to be able to drive, or I expect to be able to drive my vehicle around San Francisco, but I also expect to be able to take it to the South Bay, I expect to be able to take it to LA as well. I don't think that the future we're headed into will be set up in such a way.

For example, what I do actually think is that there will be multiple fleets that handle different use cases separately. I might take one transportation service for getting around in and out of San Francisco, I might take another one for getting to all the way to LA, but like a Hyperloop, or automatic bus service that just drives in the highways, and I might take something completely different for getting to the South Bay, or maybe some places like an hour away.

[0:08:16.3] JM: There's a lot there. Let's start with what you said about working on navigation and mapping software at a couple different places. From the software engineering standpoint, what are the difficult problems there? Is it data gathering? Data engineering? Data integration? Data pipelines? Give me some of the engineering stack of that challenge.

[0:08:40.5] RP: Sure. I can dive a little bit more into the engineering challenges that come with routing. From a technical perspective, routing has been more or less solved for the last, I'd say 50, 60 years with Dijkstra's algorithm. That's more or less what we use even today to solve every routing problem you see. It's a little bit adjusted. We use heuristics and different methods to guide the search and make sure we're not exploring parts of the graph that we definitely don't need to, but more or less it works the same way, where you start a search from either the origin or destination, sometimes both, and you expand out your search until you reach some termination condition.

The big technical challenges come with more specific use cases, like do you want extremely low latency routes, or do you want more flexible routes and figuring out how to balance these two requirements. For example, there's a open source software module called OSRM, Open Source Routing Machine, which is built heavily for the use case of extremely fast low-latency routes, but at the cost of not being able to really have as much flexibility as you might if you didn't pre-process the graph as much.

The way OSRM works basically is by taking some base map data, usually from something like OpenStreetMap and then heavily pre-processing it, basically finding a bunch of shortcuts and calling edges that don't need to be used in the final graph and adding a lot of shortcut edges in to make graph surges extremely fast and extremely efficient.

There's a problem here, because when you want to add something like real-time traffic, you can't do so very easily without having to pre-process the entire graph again. For graphs that are the size of something like North America, this can take upwards of 20, 30 minutes and you can see how that might be an issue, right, where if you take 30 minutes to pre-process your graph, then your traffic is going to be obviously out of date.

One of the big technical challenges is figuring out how to do real time pre-processing while keeping latency low. There are a couple of different ways you can approach the problem and there a couple of different techniques you can use. One pretty common one is basically partitioning the graph into various cells and being able to intelligently figure out which cells need updates based on the traffic signal you get in. Only having to pre-process particular sections of the graph and then creating shortcut edges across boundary nodes in every cell, and then doing a overlay routing.

If you want to route from here to LA for example, you might use your cell to get out and then you route cell to cell, and until you get back to – until you get into LA and then you dive in and figure out the individual routes inside and you use the cached shortest paths between cell boundaries as you move along from SF to LA for example.

[0:11:48.9] JM: You've given a perspective that you said that these mapping pieces of software have gotten rewritten from scratch a number of times. Today, we use Google Maps for lots of our navigation, like I use Google Maps for most of my navigation in the physical world, but I have noticed lots of mapping startups that are still getting started. What are the opportunities in mapping? Are these companies that are competing on redundant feature sets, or is there a greenfield of opportunities in mapping that I wouldn't know unless I was taking a good hard look at this space?

[0:12:30.4] RP: Yeah, it's a great question. From the outside, it does seem like Google Maps would offer pretty much anything you need. For most use cases regarding navigation for human drivers, I would say that's more or less – that can be accurate. I think the problem comes down to when the vehicles or even just the agents you're routing around have specific maps they work off of, or they have extremely limited capabilities.

For example, if you're an AV company and you want to come up with some fixed route, or even a set of dynamic routes to pick from for a vehicle to traverse on, you could ask Google for a path from origin destination. Google Map’s route is specifically built for humans. The route that Google gives you might actually end up being pretty untraversable in a lot of locations for an autonomous vehicle.

For example, there might be restrictions around what maneuvers your vehicle can make, there might be restrictions around what kinds of roads your vehicle can go on. For an AV company, there's a big incentive to figure out mapping and routing technology for your autonomous vehicle in particular, given the capabilities your vehicle has.

At this point actually, a lot of this is sometimes even manually done. An autonomous vehicle might be driving along a route and there's literally like a tell operator who if the vehicle has to shut down will go and manually draw another route for the vehicle to follow. Actually having solid fleet routing technology is extremely useful and extremely beneficial in helping scale up from not just having one or two autonomous vehicles on the road, but dozens and then eventually hundreds and thousands.

There's a lot also to do with lane level routing, which is something that Google Maps by itself doesn't offer exactly. This would be around solving problems like, well, even though technically a car could maybe turn right onto a street and then within 500 feet go into the far left lane to make a left turn transition. The capabilities if my AV are limited and I can make that many transitions safely without endangering either the safety of somebody walking around, or the person in the vehicle, or the vehicle itself.

Having, or being able to think about the graph and the road network at the lane level and being able to characterize transitions that happen to the lane level can be really, really useful for being able to come up with optimal, and honestly sometimes just traversable, or safe routes for a lot of these autonomous vehicles.

[0:15:06.4] JM: Is that more of a real-time challenge? If I'm trying to figure out what lanes I should be in, that seems more contextual based on what other cars are on the road around me.

[0:15:16.5] RP: Sure. I would break the problem down into two separate things. One is the, like some fleet intelligence is giving you the lane transitions to make. That might not be as granular as in 10 feet, go into the middle lane and then 15 feet later go back into the right lane. I would agree with you that that's more of like a hyperlocal decision.

At a more global level, if let's say it was impossible for your vehicle to make three lane transitions to get into the far left lane and then get into a place to make a left turn, you shouldn't have routed the vehicle there in the first place. That's a decision that we could have prevented you from making had we known that earlier. That's not something that you can use hyperlocal routing for, in the sense that even if you're able to understand that your vehicle needs to make those transitions, you may not be able to make those transitions as safely or as quickly as as you want to.

[SPONSOR MESSAGE]

[0:16:26.9] JM: Citus Data can scale your PostgresSQL database horizontally. For many of you, your PostgresSQL database is the heart of your application. You chose PostgresSQL because you trust it. After all, PostgresSQL is battle-tested, trustworthy database software.

Are you spending more and more time dealing with scalability issues? Citus distributes your data and your queries across multiple nodes. Are your queries getting slow? Citus can parallelize your SQL queries across multiple nodes, dramatically speeding them up and giving you much lower latency. Are you worried about hitting the limits of single node PostgresSQL and not being able to grow your app, or having to spend your time on database infrastructure instead of creating new features for your application? Available as open source, as a database, as a service and as enterprise software, Citus makes it simple to shard PostgresSQL.

Go to citusdata.com/sedaily to learn more about how Citus transforms PostgresSQL into a distributed database. That's C-I-T-U-S-D-A-T-A.com/sedaily, citusdata.com/sedaily. Get back the time that you're spending on database operations. Companies like Algolia, Prosperworks and Cisco are all using Citus, so they no longer have to worry about scaling their database. Try it yourself at citusdata.com/sedaily. That's citusdata.com/sedaily.

Thank you to Citus Data for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:18:11.9] JM: I want to come back to what you said about the local sets of self-driving fleets, or even ride-sharing fleets. The fleet question that always comes to my mind is how much is it winner-take-all? What are the economies of scale, or does it make sense to have smaller distributions of fleets? Do you think these are going to be large set, like large companies, like just a duopoly like it is with Uber and Lyft today, or do you think it's going to be a multitude of different players, like you have with taxi companies in New York where there's a wide variety of different taxi companies?

[0:18:50.9] RP: Yeah. For a while, I would have probably thought that this is definitely going to be a monopoly, or a centralized thing. Uber is just going to take over and Lyft is going to take over, or maybe they'll both be there. I think from what I've been seeing, the number of various modes of transportation and fleets that already exist, especially in urban centers and outside of urban centers as well are increasing a lot.

I do think it's going to be more like taxicab companies in that sense. Even just thinking about the ways that I get around this city. Yeah, I could take an Uber, I could take a Lyft, but there's also transit. Governmental authorities are definitely going to want to get into autonomous buses. There's things like Scoot, there's things like Bird. There's three or four different, I think scooter companies, not like Vespa style or electric scooters, but the ones used to – the ones you drive on, or the ones you can stand up on.

I think the number of – maybe it's because the actual vehicle types are so different, are vastly different between each fleet that it's in their company's best interest to focus on one or two different vehicle types and focus on the challenges that come around routing and sleep management for those vehicle types, instead of trying to take over everything. At least that's what I've seen in the Bay Area, right? You can maybe add some color to that as well having lived here.

[0:20:17.9] JM: Yeah. I think I agree with what you say. Another thing I think about sometimes and I don't know if you want a comment on this, but the degree to which Uber and Lyft have moats. As we've seen – with Austin. You and I both lived in Austin for a time. I don't know if you saw, but there was a time when Austin banned Uber and Lyft and then it was overnight, three or four ride-sharing companies with janky software were all stood up. I used some of them and they weren't great, but for the most part they actually did the job of Uber and Lyft pretty well.

[0:20:56.7] RP: Right.

[0:20:57.5] JM: It made me wonder what do Uber and Lyft have that is proprietary, that is a defensible moat, that a company like Waymo would not be able to subsidized a way if they wanted to. If Waymo decided, we're going to get into ride-sharing today, it seems they could just stand up Waymo ride-sharing, subsidize it more aggressively than Uber and Lyft do, take the market of all the drivers and then make the economics work, if the economics can work today. I don't know, what do you think about that hypothesis?

[0:21:32.2] RP: I would argue that I think getting to an Uber or Lyft scale takes quite a bit of engineering effort. It's one thing to set up a ride-hailing service in one city at a time, but setting it up across hundreds or across countries is a completely different question. I do agree that you can get something going for sure, and that I think if Waymo, and I'm assuming that that's what they're doing in Phoenix, or their next place that they're going to start doing some ride-hailing service.

That's actually a place where I hope that we can even eventually partner with a company like Waymo, or companies like Waymo, instead of having them try to vertically integrate actually being able to use rideOS as a layer in the sense to have us handle the ride-hailing aspect, the fleet management and the fleet routing aspects, which are really difficult problems and really different from even what Uber and Lyft handle today. I think there could be a lot there that yes, you can make that argument I think for any company, right?

We can say, “Well, why can't Google do this exactly?” I think it's a matter of focus and to some degree agility, if one company is focused on it and that's their entire focus, versus handling a lot of things at once.

[0:22:48.7] JM: Perhaps, the operational expertise and the pricing models, or even just the infrastructure for having pricing models might be a moat for Uber and Lyft. All of the software that is filling in the blanks for building larger fleets and for building internationalization and for being able to calculate maybe the costs of taking on a small autonomous pod, or autonomous scooter, versus taking a car, maybe that software will be a moat.

[0:23:21.4] RP: I think so. I think so.

[0:23:22.9] JM: Be interesting to see.

[0:23:23.9] RP: There's something also – I was just going to mention there’s something could be said also about having a layer where various companies can interact with a ride-hailing platform, and a big one is being able to share data and share things like fleet positioning. If you think about for example fleet routing versus what Uber is doing nowadays. If you're an Uber driver, you are heavily optimizing for your own profit, which is fine and that you’re incentivize to do. You'll position yourself in such a way that you can make the most money.

If you own a fleet of let's say self-driving cars, or even if they're not fully self-driving, if they're driven by safety drivers but you're dictating where they go, there's a really big problem of how do you figure out optimal supply positioning, predictive supply positioning so that you don't have – not that you want to put all your vehicles in one place. You actually want to be able to spread them out and increase utilization across the network. There are actual incentive for you to do things, like minimizing network congestion and being able to increase utilization of all your vehicles at once.

[0:24:31.4] JM: Now, I don't notice that as much in San Francisco, because it seems like wherever I am, the rides come pretty quickly. Well, I guess if I take pool, or if I take express pool, sometimes it takes a pretty long time, and that does make me think there's probably some margin to be cleaned up in terms of getting the distribution more correct there.

[0:24:53.4] RP: There's also a lot with regards to the size, or the size of the fleet for something like a pool or for Lyft Lines, right? For the most part, if you have such a large fleet, you'll be able to get something within five or 10 minutes in San Francisco, for example. That doesn't actually work across the country for sure and it's not the same. There are lots of places where pool, or line is not even available for that reason that can make the economics work.

[0:25:22.1] JM: Yeah. Or places where the urban density is more variable, that must make it much more complicated.

[0:25:28.7] RP: Oh, yeah. For sure.

[0:25:29.8] JM: Talking about the stack of technology in a car or in a semi-autonomous car, there are fully integrated approaches to building new cars. There's the Tesla, or Apple approach where you are building the entire car, and then there are less integrated approaches like comma.ai, where you are tacking on an open source piece of software and hardware to your Prius. Then there's the Waymo version, which is probably somewhere in between. What are the pros and cons of the different approaches to integration?

[0:26:08.7] RP: I think the answer is similar to pretty much any industry. With vertical integration. you get a little bit more control from the company side on the software you're going to run on the hardware specifications requirements, you can optimize for a lot of that. I think that there is a big benefit in being able to horizontally integrate when there's a big platform opportunity. From our perspective, we see that as I think if companies are able to share, or leverage each other's data in a reasonable fashion.

For example, if you're a smaller company but you come on to a ride-hailing platform like rideOS and you're able to leverage the insights of supply positioning, or traffic data, or collision data from other companies, that's beneficial to you, than having to do it from the ground up again. Yeah, I think in any of these cases, there will always be companies that I'm already invested enough to warrant continuing the vertical stack integration. I think the big benefits come around when you ask the question if like, is it worth me trying to handle every aspect of fleet management, for example.

I think a lot of these companies, especially Tesla, or even Waymo are and should be focused on making sure they get one vehicle able to follow a route, or able to go where it needs to go. Well, I think there's a lot of problems there that still need to be solved, but I think there's also similarly an equivalent number of big problems that need to be solved when you're not just telling one vehicle where to go, but telling 50 vehicles how to be optimally dispatched and where to go at once, in order to minimize cost of operations, or be able to do the tasks in an appropriate amount of time.

[0:28:00.5] JM: When I talked to George Hotz from comma, he painted a picture of the car world being much like the smartphone world, in that you would get the highly integrated Apple-like, iOS-like Tesla situation, versus the open source comma.ai version of the world. I think it's a very rigid mapping that he draws there, and I'm not sure how that will play out.

Maybe you have some opinions on that, but I think you could also imagine rideOS as almost like a Windows version, where with rideOS I think what you're doing is it's closed source, it's like AWS. I think you have some open source components perhaps, but it's AWS, or it's like Windows where you're open, you integrate with a lot of different people but you also have secret sauce so it's not exactly open-source, it's not Android. Do you have a mapping that you can draw between the car world and our historical examples, or do you think that all of those kinds of analogies are a little too much of a stretch and that this is very greenfield?

[0:29:19.9] RP: I think in the current environment, that's probably accurate, or at least the way – I'm not a fan of being compared to Windows, but that's a different right there.

[0:29:29.2] JM: Hey, Windows is a good operating system.

[0:29:33.0] RP: I think what you what you mentioned is correct, where the idea is we have some secret sauce, some ways of developing our routing tech, how we solve things like VRP, or fleet routing. We want to integrate with various partners because we believe that the data sharing, and even eventual communication between partners will be really important in the next-generation transportation system.
I think I like the way George Hotz mentioned it in the sense of there is – you can think if the car as almost like a smartphone and there are different components that come together in it. We're seeing that even with open source frameworks like Ross, which a lot of autonomous vehicles are using to manage the hardware and software integration. It's like a middleware set of frameworks for cars specifically. Well, not cars, but just robots in particular, but it's obviously used in autonomous vehicles.

I think that there's probably going to be I mean, I guess that's what we hopefully would want to be, but in the cloud, right? rideOS is the operating system for fleet management, the operating system to make a bunch of – it's not really an operating system in that sense, but it is going to be a collection of modules that you can pick and choose from to route to manage your fleet software, to manage your data, to hopefully share data with other providers as well. Because if there's a world where we have so many fleets, I think not having them be able to leverage insights and data from each other would be a not-so-great thing.

[0:31:10.5] JM: Let's get into rideOS. You are building marketplace and mapping technologies for different customers who need that technology. Give me a high-level picture of the business and the problems that you're tackling today.

[0:31:23.7] RP: Sure. From a high-level, we've noticed a lot of investment and a lot of startups coming in that are working on what we call a lower stack of next-generation transport, or autonomous vehicles in particular. A lot of companies are focused on making sure their one car is going correctly and goes to the right place.

Companies that are a little bit about, like that deep map for example are looking at problems that are related to how do we build maps from the ground up that these autonomous vehicles can use, or how do we take data from the autonomous vehicles and use that to create maps. We are operating at a slightly higher level than that, what we're thinking about is, okay, you have a couple vehicles on the road, and now you want them to start doing things. You want to actually start making money, or you want them to go complete tasks. You want them to collect data. You want them to deliver packages, pickup people. How do you do so efficiently? How do you do so optimally? How do you make sure that your vehicles will actually be able to be deployed and do what you ask them to do?

Simulation is a big part of what we'll eventually offer as a product with a lot of our integration partners too, right? Actually being able to see, okay I have 10 vehicles on the road and they can operate in this limited space. Can I do anything with this? Will this actually make me money? Is this a worthwhile investment for me? Yeah, and at the core, we have these big modules like ride route and ride plan, which are solving various kinds of constraint-based routing problems, as well as things like iterations of problems like VRP, the vehicle routing problem.

[0:33:12.1] JM: There are so many different car-related companies that could potentially want to integrate with you, but the first one that comes to mind are like legacy car companies that are very good at producing cars, manufacturing cars, getting cars on the road like Ford and General Motors, etc. What's the integration stack for a partner like a traditional automotive manufacturer?

[0:33:42.6] RP: Yeah, and just to be clear, I think you're absolutely right. One of our big targets is going to be OEMs, and even newer pure autonomous vehicle developers. Basically, the integrations that comes down to can we build modules for you directly on your vehicle, or so in the form of an SDK, or can your vehicle connect to the internet and had one of our server-side APIs?

It's pretty simple. The integration process at this point takes roughly two to three weeks. We usually have one or two people acting as integration engineers who actually dive into the code and we might have to build a couple of bespoke modules, or components that can make sure that the vehicle can actually talk to our services, upload data, request routes, etc.

[0:34:33.1] JM: Well, I was just asking about the integration process. As you mentioned, you have the ride SDK, and anybody who's a software developer has worked with an SDK, like I've worked on Android software before and there's an Android SDK, or Java software and there's an Android – Java SDK, and the software development kit lets you develop software that integrates with the platform. The ride SDK for cars, how is this integrating with the car? Is it through the CAN bus, or is there some other integration stack on the car?

[0:35:10.0] RP: I see. I see. It depends. To be clear, I don't think our components at this point will be messaging through the CAN bus. There's not really going to be pieces that we're responsible for on like, for example actuating the brakes, or acceleration. Even though we might integrate with a comma.ai-like device, we won't perform the functions of actually sending messages on the CAN bus like comma.ai might.

For example, one framework we can integrate it with pretty well and we have modules for is Ross. A lot of these autonomous vehicles have Ross as part of their software stack and hardware stack. We have specific modules that they can actually just use directly, and I think we have some different language bindings, but they would actually integrate that way through Ross. At this point, we're not entirely sure which other – what other software we would integrate with directly, if it's not Ross-based, if it's bespoke, or custom and we'd have to write some Java libraries. We might have our C++ libraries when I do that.

We're also trying to keep integration as much as possible to hitting our API, for example. The SDK would involve a collection of Ross modules and maybe some other custom language modules, but as much as we can, we'd like to have the ability to host a service, or an API that the car can head over the internet and be able to get routes or whatever they might need. As their infrastructure, changes as their software changes, that doesn't – it doesn't impact our ability to deliver them insights.

[0:36:46.7] JM: You're more concerned about the market, the ride market API and the ride nav API. You want to be able to do things like fleet management and fleet prediction and hey, get these cars to this area, because we're predicting some demand in riders that's coming up at 6 p.m., and then here's the navigation software to get there.

[0:37:11.6] RP: That's right. Now there is a bit of – for example, if a car goes offline and it needs to still follow a route, we can't just not deliver the route to them right in that case. We had to think about things like offline map caching, or even route caching and being able to realize that, or figure out what to do in situations where network connectivity may not be so good. There's a little bit of interplay there. Yeah, for the most part we are thinking about the relation in terms of you had our API and we'll provide you insights that way.

[SPONSOR MESSAGE]

[0:37:55.5] JM: We know that effective learning is active learning. When you master concepts by solving fun and challenging problems yourself, that's active learning. With brilliant.org, you can actively solve problems and have an interactive experience that can enhance your computer science expertise. Brilliant offers foundational courses in algorithms and artificial neural networks, where you can get up to speed on AI. For people with some background in algorithms, there are advanced courses on machine learning and computer memory. Coming soon is an advanced algorithms course, which will help you prepare for technical interviews.

Go to brilliant.org/sedaily to create an account for free. You need a toolkit and a framework to tackle new problems. Brilliant.org has a well-designed, colorful user interface and each lesson is packed with guided problems, with explanations and interactive quizzes.

To support Software Engineering Daily and learn more about Brilliant, go to brilliant.org/sedaily and sign up for free. Also the first 200 people that go to brilliant.org/sedaily will get 20% off the annual premium subscription. If you're like me and you're always looking for new learning materials, especially for complicated subjects like machine learning, then you might enjoy checking out brilliant.org/sedaily.

Thank you to Brilliant for being a new sponsor.

[INTERVIEW CONTINUED]

[0:39:35.1] JM: The stack for getting a route to a car, there's a series of steps like I think it breaks down into the ride plan and the ride route. Can you talk me through the process by which you generate a navigation plan and a route for a car, and what's the difference between a plan and a route?

[0:39:57.4] RP: I think maybe the slides on our website show one possible integration of like, you use a ride plan to get an overall plan for your fleet, and then it's calling in to ride route to get these individual legs. I'd like to point out, or stress that this is just one possible way of integrating. For example, if you did not want to think about, or worry about fleet management at this point, you're just not ready, or you only have one vehicle, but then so you only want to use our routing engine. We have APIs that just hit the routing engine with your vehicle-specific constraints, or map data, or both.

Ride route for example is just a layer on top of our routing engine, more or less, and at any of the services that go into getting a route for one vehicle to go from one point to another. Ride plan is more like I might have some state of the world. I have five or six vehicles and I have some tasks for them to accomplish. For example, I have to go pick up three people and I want to minimize some custom cost function. I want to minimize the amount of passenger discomfort per person as much as possible. What should my overall set of routes be? Ride plan is a layer that talks into, or that communicates with all of our internal services that help calculate an overall vehicle plan, not just for one vehicle, but for dozens if need be.

[0:41:28.8] JM: In order to generate these routes for the cars and also to control the understanding of the market, you need to have data. You need to get real-time from the car fleets, the mapping data, the traffic data and other data. You also need the mapping data to be updated on a regular basis. You of course have the – I know the historical example and I guess the contemporary example as well of these cars driving around like the old Google cars that would drive around and to constantly take maps of the real world in order to create Google Maps, how do you get that data and how does that data make its way into rideOS?

[0:42:10.4] RP: For map data in particular, a lot of these autonomous vehicle companies, even OEMs that are experimenting generally have some restricted set of map data that they use. They've built some, maybe even by hand some lane level data. Sometimes it'll be a little bit more like we can take these sorts of roads and this general whitelist zone and we might use a different provider something like OSM, or something else, to get our base map data that way.

The difference is that if they want to route on a very specific set of map data because they might need for example exact UIDs per routable edge, then we can actually integrate that into our software and provide them that. At the same time, they're hitting usually an end point for something like vehicle positioning. As the car is driving along, they're uploading their current GPS point.

We use techniques like map matching to actually figure out where like given some noisy GPS profile, where the vehicle actually is on that map data. We can actually use that to also construct things like traffic data, and that traffic data doesn't necessarily need to be coming from only one source. You can be coming from multiple sources, multiple providers, even external traffic data providers, we can use all of that to create an actual, or realistic profile of the road.

[0:43:33.7] JM: Tell me some about the, or to what degree you can talk about it the data engineering process; so the data ingest process, the processing, the machine learning model management. What's your stack there look like?

[0:43:46.4] RP: At this point, we are not necessarily working on leveraging insights from things like for example, the camera and being able to do stop sign detection, or collision detection and then using that as a signal into the routing engine. That's an eventual piece of the pipeline, but at this point it's more around being able to take for example noisy GPS and construct things like speed profiles from that data.

We use things like map matching and hidden markup model techniques to basically probabilistically figure out where the car had been most likely given some very noisy GPS trace and be able to construct speed profiles from that data.

[0:44:32.5] JM: Are you on AWS? Can you tell me some of the managed services and the data pipeline tools that you're using?

[0:44:38.0] RP: Sure, sure. We're right now using Google cloud provider. We're using some of the internal Google tools, big table and GCS for file storage. Our entire infrastructure is using GRPC. We're communicating our communication protocol is protobuf.

[0:44:56.3] JM: Are using data flow, BigQuery, any of the next generation data –

[0:45:03.2] RP: We're using BigQuery.

[0:45:04.3] JM: BigQuery, yeah. Not data flow?

[0:45:06.2] RP: Not that I know of.

[0:45:07.2] JM: Okay. It seems like, I don't know, maybe this is this is just some bias from the companies I happen to have talked to, but it seems like the more recent a company has started, the more likely it is to – and the more it leverages data engineering, the more likely it is to be on Google cloud. I mean, since you worked at Uber –

[0:45:28.0] RP: Versus AWS.

[0:45:28.6] JM: Versus AWS, it seems like the newer companies, or the people who have grizzled experience with AWS, or experience with AWS and Google cloud, they seem to be picking Google cloud.

[0:45:40.0] RP: Yeah, that's a good point. I've noticed that a little bit too. The tooling is, I think AWS and GCP both provide a pretty good tooling around managing cloud instances and stuff. I think that at least for a startup of our size, that GCP is generally a little bit cheaper. Don't quote me on that, because I might totally be wrong. I think that's why we picked it at least from what I remember. We were actually using some AWS tooling and we switched over for that reason. I think there's also some –

[0:46:10.9] JM: For cost.

[0:46:11.8] RP: Yeah, for cost.

[0:46:13.1] JM: Wow.

[0:46:14.1] RP: Yeah. I think there's also some better tooling around managing Kubernetes-related stuff on GCP. I mean, honestly, I have a feeling it has more to do with what the person in charge decides that day, right? You know what I mean? When you start a company like this, I look at a lot of the stuff we've built and a lot of the stuff I've build personally and I think like, "Yeah, I could have used this, or I could have used that framework. I could have used this language instead of this one, but we just did this because that's what we were used to."

[0:46:45.8] JM: Yeah. There are lots of different vehicle systems that could use this technology. You have food delivery robots, you have scooter companies, you have car companies, you have drones. By the way, I talked to Airwear a while ago and I think they're also using Ross, the robot operating system. I guess that is pretty widely deployed. Are you focused on any specific vertical – are you focused specifically on the car vertical right now, or do you feel like this software is widely applicable to food delivery robots and drones and so on?

[0:47:25.0] RP: Yeah. I think we're probably a little bit more focused on the autonomous vehicle vertical just at this point because of our industry experience. Yeah, that's a good insight. All of the stuff that we are working on is I think widely applicable to a non-AV use cases as well. I think that the actual – the type of problem from a technical perspective really changes when that happens.

For example, routing – being able to establish something like an UberPool-like service, or a Lyft Line-like service with some vehicles on the road is one thing. You can solve that problem in a number of different ways. When you want to say something like, "Okay, I have 50 delivery drones. They can fly around the city and deliver packages. They need to deliver a 1,000 packages within some four-hour window." There's absolutely no cost depending on if that package were to arrive earlier in the four-hour window, or later. As long as it comes into the four-hour window at some point, as long as it gets delivered, that's all I care about. All I want to do is minimize the amount of traversal cost.

That's actually a fundamentally different problem, but we're also working on problems like that. It's interesting, because it is widely applicable, but they're also like this notion of fleet management, or fleet-based routing is so wide and varied and different for various use cases that it's not like you can just take UberPool for example and apply the exact same algorithm to drone delivery, or scooter placement, or figuring out something even – something like how should I route my – yeah, how should I route my delivery bots to make sure the food doesn't get cold, but gets delivered on time at the same way as like?

[0:49:14.8] JM: Yeah, so you're looking for the problems that are highly generalizable. You talk about demand prediction for a car marketplace. That can be generalized to all car providers that want to manage fleets. You talk about navigating a car from point A to point B. That is something that is generally applicable to all cars. If you wanted to get in the food delivery business, you would probably have to figure out all of the data points for a given restaurant.

We did a show with DoorDash recently and they have all these different factors for matching a delivery person with a potential order that's in the queue, and it takes into account the restaurant and how long the food takes to make at the restaurant, and all of these domain-specific signals. It's not like you just be able to port the rideOS software easily to solve all of DoorDash's problems.

[0:50:10.5] RP: It doesn't necessarily – I think the problems that we're solving with vehicles don't necessarily, or sorry cars don't necessarily map exactly to the ones with DoorDash, or food delivery. I do think that they all are encompassed by fleet management. This is what I guess I mean by the kinds of problems we're focusing on are not necessarily only limited to routing vehicles around.

I think when you have any system where you are taking part of the autonomy that was agent-driven, where for example of you have cars that are driven by humans that have their own desires, their own wants, that's a very different thing than if you have delivery bots, or a swarm of vehicles going about and accomplishing a bunch of tasks. There's a whole host of new problems that come with just how do you optimally route a fleet, not just how do you optimally route a car, and abstract away any notion of fleet data in that case.

[0:51:07.7] JM: If you're working with a variety of OEMs, you can potentially leverage the data from all of those different OEMs if they're willing to share it with you. Is there any concern from the OEMs that this – I think the optimistic case from the OEM’s perspective would be that we're going to get to take part in this data sharing, this economies of scale and we're going to benefit from the data of the other OEMs if we give our data. The more pessimistic view might be well, if we control enough of the market, we don't want to be giving away that data, because then we're giving a competitive advantage to other people. What's the conversation around the data sharing and the economies of scale of the data with the OEMs?

[0:51:52.9] RP: I think that actually OEMs are not as adverse to data sharing as we might have – I might have initially thought too. A lot of them already do something similar with companies based around traffic congestion, like in Rick's or TomTom, where the contracts generally go around like, “I'll share data with you if you use that to make the traffic data better.” I think OEMs are more – it would be one thing for example if we said we're going to take your data and anybody, any other client on our platform can see where this data is coming from and who it belongs to and use it however they'd like. I think that would be something that they'd probably be a little bit adverse to.

If we approach it from the standpoint of your data is going to help make the system overall much, much better. By contributing, you also get access to a number of different company’s data, and in terms of you don't see, or they don't see who's providing the data. You just benefit from the insights. Your car happen to detect a collision somewhere, and so everyone got to notice at the same time, or everyone got to see at the same time that there's a collision there, and routes automatically route around that collision, for example.

I think being able to frame it in terms of your data is going to remain private, yet you can all benefit from the insights is a really important, or really – it's a conversation that OEMs are willing to have and are looking forward to.

[0:53:28.6] JM: We've mostly sidestepped the conversation around autonomy. I would love to close with getting your perspective on where we are in the journey towards autonomy and what are the big signals that you're looking for, or the big checkmarks in the timeline that you think will be significant milestones towards signaling that we are making progress on autonomy?

[0:53:52.3] RP: Yeah. I think one of the big things is also not just cars being able to drive in the city and then drive in pretty difficult circumstances, but also getting governmental support. Being able to have a city say, “You know what? Lyfts, or Ubers, or Waymo, you can start driving all throughout this one area.”

This may not even happen by the way just in the US. There's a lot of international tryst in getting cities up and running with autonomous vehicles. A lot of cities are also thinking about how to do things like launching autonomous bus services, or figuring out well, what are the costs and impact, or benefits of having established autonomous lanes?

I think when there's a also a good amount of governmental or city or public interest in the sense of cities investing in things like autonomous lanes, basically infrastructure for autonomous vehicles, that'll be a big turning point in my opinion. That might take some time. I also think for example that we won't be in a place where there's going to be fully level five autonomous for some time. I think we should be – or we'll probably be in level four vehicles for a good number of years, while data collection improves, while the algorithms improve around making sure people are safe, and that in general people are comfortable with the idea of autonomous vehicles roaming around.

Because I think one of the big things at the end of the day is even if you can – even if you were to prove, or show that autonomous vehicles on average are going to be safer in a lot of circumstances, on a lot of maneuvers, there has to be a public acceptance of the fact that there will be situations that are unsolvable. In the same way that we are now comfortable with planes being more or less fully autonomous after takeoff and before landing, and even if crash has happened, we are okay with autonomy because of just how much safer overall it is.

I think once we I guess have a public sense of this is an order of magnitude safer than it would be for a human driver, so any accident that may happen, like I'm willing to take that as like, that will happen sometimes, but at least it's better than the 10 more crashes that would have happened if humans are driving.

[0:56:23.9] JM: Rohan, I want to thank you for coming on the show. It's been really great talking to you.

[0:56:26.8] RP: Sure. Thank you, Jeff.

[END OF INTERVIEW]

[0:56:31.3] JM: If you are building a product for software engineers, or you are hiring software engineers, Software Engineering Daily is accepting sponsorships for 2018. Send me an e-mail jeff@softwareengineeringdaily.com if you’re interested.

With 23,000 people listening Monday through Friday and the content being fairly selective for a technical listener, Software Engineering Daily is a great way to reach top engineers. I know that the listeners of Software Engineering Daily are great engineers, because I talk to them all the time. I hear from CTOs, CEOs, Directors of engineering who listen to the show regularly. I also hear about many newer, hungry software engineers who are looking to level up quickly and prove themselves.

To find out more about sponsoring the show, you can send me an e-mail or tell your marketing director to send me an e-mail jeff@softwareengineeringdaily.com. If you’re a listener to the show, thank you so much for supporting it through your audienceship. That is quite enough, but if you’re interested in taking your support of the show to the next level, then look at sponsoring the show through your company.

Send me an e-mail at jeff@softwareengineeringdaily.com. Thank you.

[END]