EPISODE 1050

[INTRODUCTION]

[00:00:00] JM: Facebook applications use maps for showing users where to go. These maps can display businesses, roads and event-locations. Understanding the geographical world is also important for performing Facebook search queries that take into account a user's location. For all of these different purposes, Facebook needs up-to-date reliable mapping data. OpenSteetMap is an open system for accessing mapping data. Anyone can use OpenStreetMap to add maps to their application. In some ways, it's like a Wikipedia for maps. The data in OpenStreetMap is crowdsourced by users who submit updates to the OpenStreetMap database. Since can anyone can submit data to OpenStreetMap, there is a potential for bad data to appear in the system, just like sometimes you read wrong information on Wikipedia.

Facebook uses OpenStreetMap for its mapping data, and this includes important applications where bad data would impact a map user in a meaningfully negative way. In order to avoid this, Facebook builds infrastructure tools to improve the quality of its maps.

Saurav Mohapatra and Jacob Wasserman work at Facebook on its mapping infrastructure, and they join the show to talk about the tooling that Facebook has built around OpenStreeMap data.

[SPONSOR MESSAGE]

[00:01:31] JM: If you are selling enterprise software, you want to be able to deliver that software to every kind of customer. Some enterprises are hosted on-prem. Some enterprises are on AWS. There might be a different cloud provider they use entirely, and you want to be able to deliver to all of these kinds of enterprises.

Gravity is a product for delivering software to any of these kinds of potential environments or data centers that your customers might want to run applications in. You can think of Gravity as something that you use to copy+paste entire production environments across clouds and data centers. It puts a bubble of consistency around your applications so that you can write it once

and deploy it anywhere. Gravity is open source so you can look into the code and understand how it works.

Gravity is trusted by leading companies, including MuleSoft, Splunk and Anaconda. You can go to gravitational.com/sedaily to try Gravity Enterprise free for 60 days. That's gravitational.com/sedaily to find out how applications can run the way that your customers expect in their preferred data center. That's gravitational.com/sedaily. Thanks to the team behind Gravity, the company Gravitational, for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:02:57] JM: Saurav and Jacob, welcome to Software Engineering Daily.

[00:03:00] SM: Thanks. Hi, Jeff.

[00:03:00] JW: Yeah! Hi, Jeff. Thanks. Glad to be here.

[00:03:03] JM: Yeah. You guys have both worked on a variety of geospatial and mapping problems. Could you give a bit of background on what your experiences are with mapping and geospatial data?

[00:03:16] SM: Sure. I joined Facebook close to 6 years back, and at that point of time, we were starting to rollout our own map stack. I was one of like the first two or three people on the map's team, and I always remind Jake that I'm the OG mapping team member. But what actually happened was we started with basically a sleepy maps interface, because remember, this is 2014, right? [inaudible 00:03:37].

We started rolling out or own mapping infrastructure and we used a third-party map, but then we realized at that point of time that something like OpenStreetMaps actually is there as a community for [inaudible 00:03:50] and growing. In fact, we did a parallel prototype with OpenStreetMaps. Then after two, three years when we felt that OpenStreetMaps has achieved a critical mass, we actually started projects to switch completely to OSM as our base map.

That's how this theme, again, started and sort of focused on basically creating the Facebook base map experience with OSM as the cornerstone of the data.

Jake?

[00:04:18] JW: Yeah. I'll just give a little bit of my background. I sort of fell into the whole mapping and geospatial work. I was working at a place called MIT Lincoln Laboratory and I was there for about 9 years or so after college. A friend of mine was starting this startup where we were building a navigation app that tried to talk to you like your friend was sitting in the car next to you. Instead of giving these instructions that was like in 200 feet, turn right on main street." They could say things like, "Go under the railroad tracks and then turn right at the stop sign."

I was really looking for a change of pace. I decided to join that, and that was in 2012, I guess. Yeah, we built that whole thing out. We launched on the app store. We're actually acquired by Verizon. I was there for a little bit. Then I found out the Facebook Boston office was doing all these work with mapping and geospatial infrastructure and location and I was super interested in that and ended up joining the team. I was able to like continue that work. It's been quite an experience and pretty fun.

[00:05:16] JM: When I think about geospatial data and mapping data, to me it's very different than the kinds of data that you store in a typical relational database or a document database or even maybe a graph database. How does geospatial data get modeled in a database?

[00:05:39] SM: Actually you're right, that there are a lot of nuances in storing and querying geospatial data. In fact, one of my first projects at Facebook was rolling out a geospatial indexing system. Like the way we index normal data, which is one-dimensional, verse a geospatial data that has latitude, longitude is slightly different and there are nuances and optimizations to be done.

I built – I was one of the first engineers on building something that's basically now a Facebook scaled geospatial index. The major thing you have to watch out for is that the same thing that you consider a perceived weakness of geospatial data, which is like it's two-dimensional and spread out, is also a strength, because the world isn't random. Things occur in a pattern.

Geospatial data is actually easier to shard. Like the whole world, if you look at let's say streets, which are line segments, you won't have like an even distributed, like three-quarters of the world won't even have streets because it's water. You have to approach it in a way that there are techniques like R-trees, Google S2, the Hilbert index, space-filling curves. You have to leverage a lot of that and bring it down to a subset of problems, which then traditional databases like the fast joins or distributed joins, MapReduce can then handle.

[00:07:04] JM: I see. You have to figure out how to model that geospatial data in a way that basically can work for traditional relational database architectures.

[00:07:15] SM: Or whatever you have. You look at this Facebook infrastructure is very build out, let's say brain scans and joins, right? We can do that at scale. We are very good at graph databases. When we started this at original sort of explorations where, okay, we have this multidimensional data but we're very good at like joining two tables on a key, like in a distributed environment.

Therefore, we made investments into like Google S2- inspired, Hilbert, space-filling curves and stuff like that. What that allowed us to do was to convert what would [inaudible 00:07:49] effect be a two-dimensional [inaudible 00:07:52] scan down into a linear key where you search within different ranges, which are the Hilbert cell IDs.

[00:08:00] JM: How does Facebook use mapping and geospatial data in its applications? What are the use cases?

[00:08:08] JW: Yeah. There's really like a number of surfaces that you'll see a map on Facebook. There's a local app, which is a way people find local stores, restaurants, things like that in their area. There's marketplace, which is sort of a way people can sell things. That has a map application on it. There's business page listings. People have seen like if you ask for a recommendation for a certain store in your area or – I don't know, like a plumber or something like that, Facebook will actually recognize that and popup a map and anyone who answers and says, "Oh! You can go to this store or that store or I recommend this kind of person." If they

have a Facebook page on it, it will actually like populate a map as people go and add these things to the discussion.

There's also like business check-ins. It's kind of like a Foursquare functionality. There's ones, just to be specific, the ones people see if you're like at an airport and you check-in and say where you're going. There's this little airport. It shows a little airplane line that shows where you're going. Our team like builds and serves all those base maps.

[00:09:05] JM: How accurate does the mapping data within Facebook need to be?

[00:09:12] SM: Accuracy is relative to the context. For rendering, right? Depending on your zoom level. There is also a sociopolitical context accuracy of things like borders, like borders are highly contentious in the way that there are [inaudible 00:09:28] in the world. In those areas, what we do is we have and we ingest map data, we render it. We actually have policies around like how to render the borders which comply with like topline policy of Facebook.

For things like [inaudible 00:09:43] and things that we show in the base map which are mapped to real-world places, depending on the use case and sort of the privacy compliance issues, the accuracy is different. What the project that Jake and I started and worked on was about given these set of constraints and top-down directives of accuracy and all of that, can we ensure that we are meeting those milestones?

[00:10:08] JM: Now, let's think about a few kinds of applications that I could think about. One is like if I make a search query on Facebook, and that query is going to take into account my location. If I'm searching for restaurants. Then maybe the mapping data doesn't need to be perfect. The geospatial data doesn't need to be perfect, because you're going to be adding the weight of location into a number of factors that go into the search, like what are my food preferences. What restaurants have I gotten to in the past?

On the other hand, if you're trying to find where a specific restaurant is on a map, then you would want the location data to be perfect. Do you have like different ways of accessing the location data in these two kinds of use cases where you need high-accuracy versus lower accuracy?

[00:11:04] SM: Yeah. You can think of it like images, right? There is an absolute high-fidelity location that we have, which may or may not map to the real-world, but it's continually improving, but that's a very fine-grained location of let's say a restaurant, a restaurant pin. For different use cases and all of these things, you can now map that pin based on reserve coding to, "Okay, it's in this zip code or this city or it's near this street." There could be faster systems, as you said, for the codes or use cases which sort of run-off of those things for rapid computation. But when you are actually like what are the restaurants within 500 meters of me? Now you need to access the high-fidelity data. This is practically how – It's a tradeoff between like how accurate and intensive you want the computations to be versus how quickly you want the result.

[00:11:54] JM: The way of building a physical map of the world, there's a number of approaches you can use. None of them are easy. I mean, obviously, historically speaking, the Google approach was literally drive a bunch of cars around map the physical space. I think in tandem with some other approaches like satellite data.

Facebook came around a little bit later and you had access to a slightly wider set of tools a little further out on the technology curve including OpenStreetMap. Explain what OpenStreetMap is.

[00:12:34] SM: OpenStreetMap, you can think of it as Wikipedia for maps. It's basically – I know, it's not quite accurate or description, but it's basically a community-oriented effort to edit and create maps of the world, including places you live in. What this produces is it's an open interface. You can go to the OpenStreetMap website and you can just start editing the world. All the changes are live.

I will come back to this. This is why the project Jake and I started sort of exists. But what this allows us to do is places that are – Like the middle of Africa or Southeast Asia. People can actually map their own communities. The data in those ideas sometimes OSM is the only source for detailed data for, let's say, like roads in Nigeria or Tanzania, because the community is so active and so wonderful. They keep on mapping and improving their own community.

In fact, initially, OpenStreetMap actually have volunteers who would be on bikes or hiking trails with GPS, like those little course GPS readers, and just point-by-point map-out these backwards roads.

[00:13:44] JW: Just to build on that. I mean, I'd say there's like two things that really set OpenStreetMap apart from any other map database you might find. The first is that, like Saurov was explaining, anybody can go in and just edit the map just like Wikipedia. If you haven't seen OpenStreetMap and you're listening right now, you should absolutely go to openstreemap.org and create an account and like zoom in on your neighborhood or your hometown and see if there are things that are missing, things that are wrong, and just learn how to edit and fix them, and those things end up in tons and tons of applications and probably ones that you use.

Really, the other thing is that anybody can download the data. It's not really even just like the map display and how it looks. It's actually the raw data that comprises the map, like the real longitude and latitude positions of all the roads or all the buildings or trees or anything else that you find in the database and then people can build really any application you want on that, and that's really what's made it just really wildly powerful and a huge competitor in the geospatial market.

[00:14:44] JM: Tell me more about how the crowdsourcing of the OpenStreetMap data works.

[00:14:49] JW: Yeah. It's very much ground-up and really community-driven. There's no real overarching sort of organization that's saying this is the type of thing we want. It's whatever any individual interested in at the time. There are a lot of these little communities and people that have their own interests and things they're interested in mapping. You'll find tons of people who are really into hiking or really into mountain biking and like sort of said they attach GPS devices to their bikes or their packs and they go out and like around they go on hikes and they come back and they can upload those traces to OpenStreetMap and then find out what's been missing? What's there? What's wrong and adjust it and fix it and add names and change, "Is this road paved or not?" and put in whatever kinds of things they're interested in.

There're a lot of these almost like groups that are interested in certain types of data that really focus on things going on that they're really interested in. There're also just people who are like

really interested in just their neighborhood and their town and making sure it's perfectly and accurately mapped. Every business that opens or closes, they make sure that OpenStreetMap is really up-to-date or there's any construction project and a new house appears. They are just on top of it and mapping it or any road change. People are just –There's really a wide variety of interest there and it's very much just bottom-up driven.

[00:16:11] SM: My favorite is people who are power line enthusiasts. You will find like this transmission power lines very carefully mapped out with kilovolt ratings and like sag and sort of tower placement. I mean, I love that you can actually go and do that if you're interested in it. People map benches –

[00:16:31] JW: Yeah. I think there's another great group of like railroad enthusiasts and people who are just obsessed with railroad infrastructure and they love its history and like not just where train tracks are now, but where they've been, and you can find just these amazing maps of old train lines that are now paved over and you can find them. I just saw one near my house near my house the other day and there's like this supermarket that has this weird hill and there's an auto body shop with this very weird shaped like parking lot and it always stood out to me as this odd thing. I saw, it's literally like a month and a half ago in OSM. The actual train line used to go through there. I had no idea. This hill was there to support to train and it was like on this curve, and that's why the parking lot goes like – The train used to go right down the middle of what's now the parking lot. I mean, it's really amazing. There are just these groups of people that have coalesced on OpenStreetMap as the place to like put and stores and maintain this data. It's really amazing.

[00:17:24] SM: Mapping has its own Maslow pyramid, right? You have areas of the world which are not mapped at all like Southeastern Asia, middle of Sub-Saharan Africa. There, it's more about like getting a map. But if you come to like Europe or North America, at least coastal North America, a lot of it has been mapped that is data available, but what OpenStreetMap allows to do, as Jake said, is to add different layers, like move up the Maslow pyramid towards like self-actualization. This is what is so wonderful about OpenStreetMap. It allows like the whole world to have a map and then refine and update that map as time goes along.

[SPONSOR MESSAGE]

[00:18:16] JM: If you listen to this show, you are probably a software engineer or a data scientist. If you want to develop skills to build machine learning models, check out SpringBoard is an online education program that gives you hands-on experience with creating and deploying machine learning models into production, and every student who goes through SpringBoard is paired with a mentor, a machine learning expert who gives that student one-on-one mentorship support over video.

The SpringBoard program offers a job guarantee in its career tracks, meaning that you do not have to pay until you secure a job in machine learning. If you're curious about transitioning into machine learning, go to softwareengineeringdaily.com/springboard. Listeners can get $500 in scholarship if they use the code aispringboard. This scholarship is for 20 students who enroll by going to softwareengineeringdaily.com/springboard and enter the code aispringboard. It takes about 10 minutes to apply. It's free and it's awarded on a first come, first served basis. If you're interested in transitioning into machine learning, go to softwareengineeringdaily.com/springboard. Anyone who is interested and likes the idea of building and deploying machine learning models, deep learning models, you might like SpringBoard. Go to softwareengineeringdaily.com/springboard, and thank you to SpringBoard for being a sponsor.

[INTERVIEW CONTINUED]

[00:19:52] JM: You talked about all these different things that are represented in OpenStreetMap; railroads, and businesses, and benches, and roads, and sidewalks. How does this data get laid out or schematized? What is the OpenStreetMap schema?

[00:20:12] SM: OpenStreetMap is designed as an interchange format. It's like they have their own database schema, which is open in a PostgreSQL database in the OpenStreetMap sort of organization. But when the data is given out, it's dumped out in a format called the planet file. What the planet file is – OpenStreetMap uses three basic sort of core data structures to represent the world. The first one is a point. It's called a mode. It has a lat-long, a latitude and a longitude and a series of tags. Tags are something will come back to again and again in OpenStreetMap. These are basically freeform spring key value pairs where you can say, "This is

node." You can say it's a restaurant, like place type restaurant, and you can say amenity, bathroom? Yes.

That's a node. That represents a single point. The next data structure that they have is called a way. A way represents an ordered lists of points. It can be anything from lines, polygons. Most of the features that we get about in OpenStreetMap are ways. Then there is this thing called relations, which allows you to map sort of – Basically think of it as a graph edge. It allows you to create an association between two different types of OpenStreetMap individual like a ways and nodes.

For example, you have a polygon with a hole in it, right? It's a building but it has like enclave inside. You can basically have two ways. One for the outer polygon and one of the inner polygon and you can say that these two are actually related in the way that they are a multi-polygon and the inner one is a hole. That can go on the relation. This is how OpenStreetMap data is laid out, which is great for interchange because all you have to do is read this three lists. But when you start doing stuff with it, this is not really a transactional schema.

[00:22:00] JM: Indeed. OpenStreetMap itself uses Postgres under the hood. I think we can start to get into your own usage of OpenStreetMap data. As I understand, your consumption of OpenStreetMap data, you can consume that interchange format. OpenStreetMap, anybody can export the data that is sitting in that planet file format and can consume it. Then you can import it and do whatever you want with that high-level representation.

When you're ingesting that planet file format, what are you doing to put it into your own database schema? What's your underlying database?

[00:22:47] JW: Yeah, that's a good question. Most of Facebook's database in terms of the data warehouse that we use is built on Presto. I know you had an episode about Presto I think like a couple of months ago or so. Yeah, we take that planet file and we populate it in a way that we can upload to what we call our hive data store. Basically, we use a schema that's pretty similar to one that people would put into Postgres but is in a Presto data format, and that's just available for anybody to query. From there, we actually have a whole map data pipeline that can take that data, filter it based on things cartographers are interested in. Slice it up and dice it in

ways that optimize it for serving maps out to consumers and those go into a variety of different indexing services and CDNs and things like that. So that way we can actually like deliver these things optimally.

[00:23:40] JM: That's kind of interesting. You don't actually keep it in a database in like – Well, at least on disk in a database. You just put it in that planet file format. You index it somehow in your hive metadata store and then Presto interfaces with the hive metadata store and Presto uses its magical memory optimization stuff to pull it into memory and perform operations on it, right? You're just doing all these operations in-memory.

[00:24:08] SM: Yeah, because a lot of the workflows of what we do with OpenStreetMap are kind of batch workflows, right? You ingest the data then we'll talk later about ingestion and integrity things. All of these need to run on a massive amount of data, and that's what Presto buys us, is the scale. Because this is not transactional, like we're not changing it. We are just consuming it. We take it, and as Jake said, we put it into higher data warehouse and then there are like these series of pipelines that run on it to create different inter-mediator in the final sort of data structures, all of it.

[00:24:43] JM: This is actually what leads to these interesting set of problems that you had to figure out solutions to. Facebook wants to use OpenStreetMap because it's this amazing open dataset of mapping information. But the way that you consume it is this batch ingest, this batch pulling it into all your different services. Kind of a batch ETL-like pipelines. That has all of the issues of batch processing. In addition, there's the issue of correctness, where when you pull this data into the Facebook infrastructure, there are certain instances where OpenStreetMap might have been tampered with or updated recently in a way that is not so good.

When I used to write reports in high school and I would use Wikipedia, sometimes you go to the Wikipedia page on animals and it lists like tofu as an animal, because somebody vandalized the page recently and said that tofu was an animal. Me being in high school and not knowing any better, I might write an entire paper on the taxonomy of the tofu species.

But that's not really a problem. Maybe I get a bad grade, but it can be much worse in the case of OpenStreetMap if somebody edits something and says there's a road leading from village A to village B and actually the road leads off a cliff. That's really bad.

[00:26:16] SM: Yeah. Jeff, my favorite example when Jake and I were initially looking at this sort of OpenStreetMap change corpus to see, like what kind of bad changes are there. Jake, you remember those lakes inside the malls?

[00:26:27] JM: Oh my God!

[00:26:30] SM: People figured out that a certain game about on-the-go capturing these animals that can evolve, like Pokemon Go. Pokemon Go seemed to use land use at some point of time from OpenStreetMap. These teenagers would go around, change, creating like fake lakes so that water type Pokemon would spawn in their building complex. They wanted to catch Squirtle, basically. We used to find like this weird sorts of land use inside like buildings, malls, schools.

[00:26:59] JW: Yeah, and to build on that, I mean, a more serious example. There's kind of a very similar famous vandalism moment in OpenStreetMap a few years ago where somebody was able to change the name of New York City to what amounted to an ethnic slur. This slipped through the radar of some integrity checks and some companies ended displaying that on their map to their users. You saw this on Twitter, people were like, "Why is, I think, like the city bike app or something showing New York City labeled as …," and it made it on like TechCrunch. It was on the cover. I think like in even The New York Times picked this up.

[00:27:36] SM: No Facebook though.

[00:27:38] JW: It was not on Facebook. But sort of like this almost like watershed moment I think for a lot of people using OpenStreetMap.

[00:27:44] SM: People, they fixed it. They fixed it very quickly, but the screenshots live forever, right? Like the screenshots, the Internet has memory with screenshots.

[00:27:54] JW: Right. It got fixed in OpenStreetMap really quickly, but then it ended up just being delivered out to all these apps and their consumer users for I think a few hours. It's kind of a moment that I think a lot of companies said, "All right, we need to do something more serious about this kind of potential vandalism," because it's more serious than a lake in somebody's apartment complex.

[00:28:13] JM: Right. In taking this data that maybe tampered with. I mean, the problem is it's the size of the planet. There's a lot of judgment to make if you want to verify that the entire planet has not been tampered with on every batch ingest. That's so much data. Now that said, you do have a well-formed set of changes that have been made to OpenStreetMap in each time you ingest it.

Like let's say you've got your local copy of OpenStreetMap that you ingested yesterday and today you're ingesting it again. You're going to be able to look at a well-defined change set. How have the crowdsourcing people made updates to the public version of OpenStreetMap? You can decide which of those changes to actually apply to your own local version.

[00:29:08] SM: Jeff, it's not that simple, and we can talk about it.

[00:29:11] JM: Yeah, tell me more.

[00:29:13] SM: Yeah. The changes, if you look at the way people submit changes to OpenStreetMap, like they will submit changes, like somebody will say that, "Okay, I'm going to fix all the labels on high schools in a district," right? The changes will spread across like a district's surface area. Somebody will say like, "Let me go and fix national park back." One day, they will attempt [inaudible 00:29:33] and just change national parks. These changes are all over the place, and sometimes the changes mix and match different kinds of changes.

If you are consuming it, as you said, as a batch. So you basically end up with the changes that are in a bunch of these change sets and the same feature might have been changed twice or thrice, right? What you're going to have is that notion of a change set is very good for curating it into the crowdsourcing sort of catalogue. But when you're in ingesting it, you basically have to deal with the individual changes, and what we did was, Jake, me, and Kevin [inaudible

00:30:13], we created an algorithm which basically took these changes and re-clustered them. We sort of created these logical change sets out of the net change that happened between two snapshots.

If you think of it, OpenStreetMap is a [inaudible 00:30:27], right? Like changes are coming in and bunches of change sets submitted by people, which consists of individual changes. When you take the difference between two snapshots, you can think of it as basically a combination of all individual atomic effective changes. Let's say somebody changed the name of a place from A, to B, to C. We just see that it changed from A to C.

We take these atomic changes, which is basically a diff algorithm, and then take these and clustered it to see that how many of these changes are interlocked so that all of them have to be reviewed together. If you have a T-junction, like a stop sign in a T-junction and somebody moved the point inside. So now the junction point, now the three road segments that connect and the junction point are interlocked, right? You cannot just change, move the point, then you'll have two roads hanging off of nowhere. You have to either accept or reject all four of them. That's the foundation of how we approached the batch ingestion at scale.

[00:31:30] JM: This batch ingestion process, it presents a tradeoff between freshness and correctness, and you've given a little bit of context as to why that is. Describe that tradeoff between freshness and correctness in more detail.

[00:31:46] JW: Sure. Yeah. It's really just like you said. People are editing OpenStreetMap constantly. OpenStreetMap publishes minutely, essentially, database replication logs. New roads really are being created all the time. People are adding buildings and things like that, updating business listings all the time. But if we want to stay fresh, we essentially have to ingest all that stuff as it comes in, say, every day or even every minute. But if we want to take the time to slow down and say, "Find the things like Saurav is talking about or find when they are cases that there might be vandalism," and we can't accept some of those changes, we sort of introduce this delay. It's really hard to stay up-to-date while also being able to evaluate all these changes and say, "Are these things correct in some way?"

What we've done through this system that Saurav was talking about is a way in which we can sort of pull in data that we feel is correct and that stuff can stay up-to-date and fresh. The pieces of the data where we're maybe not sure or maybe look like they're vandalism, we can hold those parts back and find all the other changes that are interconnected or interlocked, as Saurav was saying, and hold those pieces back while keeping the rest of the map sort of fresh in that way.

[00:33:04] SM: The key has is to keep on doing it again and again. There are some interesting mathematical properties that come out of our clustering algorithm. Each changed feature, because we are taking a net diff, each feature is in one logical change set. A logical change set is an all or nothing, like either the whole thing needs to be approved or the whole thing needs to be rejected. One bad feature or one sort of wrong thing in it, you need to reject the whole thing.

But the good thing is because these are sort of like item-potent, commutative and independent logical change sets, you can pick that up again. It will again show up next time you run a diff. The beautiful thing here is that, as you said in your Wikipedia example, somebody might come and fix it later. If by the time you do it again and somebody has fixed it, the net diff won't show it, right? It will get auto-ingested

That's the beauty of – I mean, theoretically, these are batch consumption, but it's more like micro-batching. What we are talking about is we are trying to do this on a faster and faster and a regular cadence so that the net diff is small and then our sort of the machine augmented review and user review together can capture and cover a lot of these changes.

[00:34:20] JM: All right. How often is this batch ingest taking place?

[00:34:25] JW: Right now, essentially, we evaluate it every day. But yeah, our process runs sort of weekly, like the full process that Saurav was talking about, where we take in a whole new dataset. We take what is the change between the last one and the current one and we compute these logical change sets and we have a bunch of ways, I know we'll get into later, about automatically accepting some of them, and some of them go to manual review. That happens essentially on a weekly cadence. Every week, we can publish out what is the latest map that we

feel comfortable with that has a lot of data over the past week and maybe [inaudible 00:34:58] some changes that we still want time to evaluate.

[00:35:00] SM: I mean, you can think of this as a CICD for map data, right? Continuous ingestion, continuous deployment. What this process has allowed to do us is pick up all this process from like this linear thing of ingest, verify, deploy to OpenStreetMap moves as a parallel stream and this sort of batch ingestion keeps on calculating their diffs, then reviews can happen separately and you can ship any sort of approved snapshot out of it.

It brings that discipline, which has been there in like source code control world for a long time. In fact, our sort of the logical change set algorithm and all of these are highly inspired by things like cherry picking changes when merging between two source codes. We put some restrictions on it in the way that the only place changes can be written to is upstream OSM. That means we aren't immutable subset of the things that are in OSM. We have just kept the bad stuff out. This helps us avoid like the diamond problem, right? Two folks, and then you have to rebase them.

[00:35:57] JM: Very cool. On each of these ingestion processes, there are some changes to the map that you're going to want to flag as maybe we shouldn't apply this change immediately. Maybe we should wait another couple of days. See if somebody fixes it or maybe we should put it through our own review process. I think that these kinds of changes that you identified, they are called suspicious change sets. Is that right?

[00:36:29] JW: Yeah. Yeah, we have a number of ways of looking at a variety of changes, right? We actually leverage some of Facebook's profanity detection tools and say, "Does this name look bad, or is this maybe matching some bad word?" For example, Massachusetts contains the second, third and fourth letters. They are technically a curse word. So something like that can get flagged and those things can be held back that somebody should review and say, "Does that look like a real name or is this possibly like a curse word?"

What our process does is if a reviewer actually does find something that looks wrong in some way or is vandalism, what they can do is actually just go into the main OpenStreetMap database, make that fix, and then the next time we do our ingestion process, we can make sure

that we pull their fix-ins so that way our map maintains that level of correctness and we're never actually like pulling in these types of changes.

[00:37:23] SM: An interesting thing to note here is that these usual suspects, like the word asses in Massachusetts or the city called F-U-C-K in Austria. This always reminds of that [inaudible 00:37:33] cartoon about the guy who thinks he have this plate as like [inaudible 00:37:36] and the cops will always get confused, but then they have his number on like a little sticky note. These sort of the usual suspects, we have factored them into the kind of the process itself so that if this is a recurring pattern and it has been whitelisted, it will just go through.

[00:37:56] JM: Okay. Let's talk about what actually happens when you've surfaced these changes that you are not sure if you should apply to your local OpenStreetMap. Because you suspect some profanity or you suspect that somebody is updating OpenStreetMap to put lakes in the middle of shopping malls to spawn Pokemon. Whatever the reason, you decide that you've flagged these things as suspicious. What can you do? What are the options once you have identified these things are suspicious?

[00:38:32] SM: Yeah. Remember that we treat OSM upstream as a writable copy, right? What we do is that we open tickets from mappers and basically surface it to editors who can go and make that change so that, as Jake said, next day the cycle runs, it will pick up the changes.

But some of these, one of the biggest sort of philosophical hills we have to climb was when we started this project, a lot of our validators, like the human reviewers, they came from an editing world. Editing is about quality. It's about making the best change you can. But [inaudible 00:39:06] is like a shovel, right? It's not a scalpel. It's about is this thing bad enough to keep up?

Basically, teaching our sort of the human reviewers, that you are not doing quality control at the upper-end. What you're doing, you are basically keeping the bad stuff out. Sometimes, details in geometry is one thing. It's like editors are used to creating the fine-grained detail or a road curving. But if there was no road and somebody has created a road, which is like with four points, which is kind of jagged, is that something we should pass or fail?

This kind of small things were more important in the initial phases of the project is just teaching people the philosophy that when you're ingesting, it's about volume and keeping the egregiously bad stuff out. It was like over time since we have been doing this, we're close to a year, year and a half now, the mapping team and the engineering team and the tools are not in-sync so that we feel that this is now a smooth cycle where if the machine augmented review flags us as [inaudible 00:40:09], we have a human look at it and then if the validator feels that this should be edited, they just open a ticket using MapRelate, which one of the things that you use to create OpenStreetMap pass, and they basically surface up and somebody goes and fixes it. If nobody fixes it, next time you run, it will again get flagged and not get into the map.

[SPONSOR MESSAGE]

[00:40:40] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

[00:42:28] JM: Okay. There is both automated review and human review in the process of determining which of these location changes you want to actually apply to your local copy. Could you go a little bit deeper into both the automated review process and the human review process?

[00:42:49] SM: Sure. I can cover the automated review and Jake can cover the manual review part.

[00:42:53] JM: Sure.

[00:42:53] SM: If you look at it – You know the Sansu thing, which is I think it's either Sansu or [inaudible 00:42:58] map?

[00:43:01] JM: Right.

[00:43:01] SM: Basically, there is so much nuance in what is the correct representation of the world. We can build things that can catch like the [inaudible 00:43:14], like obviously lake inside a building. We can say that that's not the [inaudible 00:43:17] approved place names. As I said, borders or other sort of things like whether something is the name changes, like sometimes schools are renamed, districts are renamed, or roads, suddenly there's a two-lane road. Somebody did a construction. It has now a four-lane road.

These kind of things actually require humans. Our philosophy has always been that we cannot have – Obviously, it will be ideal to have a fully human pipeline, but somebody did some computations, and the economics of that, it's mind-blogging, because of the firewalls of changes that OpenStreetMap is.

What we decided was when we analyzed all the changes on the machine review part, we realized there is a lot of changes that a human doesn't need to look at. You don't really need the

nuanced sort of [inaudible 00:44:06] heavy, time heavy take that a human brings on these sort of basically give me changes.

That's what the automated review takes care of. But the human changes are actually the way the humans decide what changes should go in or not. There are some changes that as of now we haven't found a very reliable way of doing with machines, and Jake can actually cover that.

[00:44:30] JW: Yeah, sure. Like Saurav mentioned, all these changes get grouped together into what we call logical change sets and it goes through this automated review process where it can look at things like does this make sense given satellite imagery data? Does this have some words in it that maybe I should have somebody look at?

Anything that it's not sure about or that it doesn't know how to handle goes into this manual review bucket. What we've done is we've built like a specialized UI that exists internally where mappers can look at these changes and what happened, and they can do it at sort of a high-level, like how does this thing actually influence the map? Is it part of some polygon that now it doesn't render? Does it move a lake somewhere? Does it look like maybe somebody was just taking the accuracy of some road and like really making it nice and smooth or something like that?

Our mappers, who are very familiar with OpenStreetMap and this process can go in and sort of inspect like, "All right, what actually happened here? What are the nodes or the relations or the ways that changed here? What are the tags that changed? Does that make sense?" It's a very simple for them. It's a binary decision. Either accept it or reject it. If they reject it, they can say, "Hey, this is flagged. This need to change on OpenStreetMap." They can say, "Well, I'm not really sure about it." So they look at those things and then all the ones that they accept at the end of the day, we can take all those changes and essentially apply them on the map while leaving the old ones behind and knowing that we haven't really broker the map in any way because have this logical change set framework.

[00:45:56] JM: All right. These collections of changes that you detect on each of the ingest processes. When you're grouping these together and you're turning them into collections, like the [inaudible 00:46:11] collections, is it kind of like when I go into a code review tool and there's

a diff and the tooling makes it easy for me to sequentially go through the different sections of the diff? Like if I'm a human reviewer and I'm reviewing the location changes, is it sort of like interfacing with code changes on GitHub?

[00:46:37] SM: Yeah, it's a visual diff tool. Like the tool that Jake talked about. It basically shows the two versions of the map, one without the changes, one with the changes with the changes highlighted.

[00:46:46] JM: Okay.

[00:46:47] SM: So that's like the first line of check. Think of it as like a diff view UI, right? As you said, on GitHub or [inaudible 00:46:52] you go, it shows you this line was removed. This line was added. These lines were edited. Now, even though the entirety of the change set might cover a long big area, you now know which ones to focus on. Remember, the machine augmented review is run before this. What it does is it highlights only the pieces that we feel that a human should look at.

[00:47:14] JM: Okay. That actually is applied after the automated review process right? You get these collections of changes and then there's some stage in the pipeline where there is automated review. This is like you've trained models to be able to review some of these changes, right?

[00:47:34] SM: Yeah. It's a mixture of heuristics, AIML, some computer vision stuff in there. It's a rule engine. It basically has a series of rules, right? The rules run one-by-one on the individual feature changes and classify them, give them a confidence of how confident are we that these change should be accepted? Then we have some post-processing, which basically says if a logical change set has all features auto approved, then basically it's good to go.

If there is one feature that's flagged as reject this feature, then that logical change set just goes down. Then that leaves us with a bunch of logical change sets of which let's say 90%, 95% have been approved, but only 5% of the features need to be looked at. These are the ones we surfaced to the manual review.

[00:48:19] JM: That automated review, the rules engine, how did you build that? What are the – Obviously you can do things like scan for certain words that people might be adding, like profanity. That's pretty straightforward. But what were the learnings that went into creating the rule engine for this automated review process?

[00:48:42] SM: We didn't built it straight up. We actually observed our – Like the manual reviewers and the validators and basically we took notes on how much time to spend on something before being confident about it.

Like say, for example, geometry detailing was very easy to support [inaudible 00:48:59]. Like a road had three points and somebody added like a smooth curve. But if you look at it from a numeric standpoint, it might have added like 200 vertices, but we then went back and tried to capture this sort of domain knowledge into rules. The geometry detailing, we decided that we can do basically a curve similarity.

Like in a simple [inaudible 00:49:22] similarity between bounding boxes, central movement. By some way, we sort of cheaply mimicked the way humans make weak decisions. We already have a project that basically generates roads out of satellite imagery, like in places like Thailand, Asia, Africa. We basically repurpose the models from that project using neural networks and we repurposed those networks to do the inverse.

If somebody has created a road, we go to the satellite photo and check, "Does this road roughly correspond visually to something that's on the ground and that looks like road?" There are like OCR-based [inaudible 00:50:02] is people actually try to write letters using roads. We check if something looks like – There is a picture of – What was it? There was a picture of [inaudible 00:50:12] America.

[00:50:14] JW: Yeah.

[00:50:15] SM: So these type of things, we use OCR. There is a lot of NLP that's used for the profanity and all of that stuff. Plus, the inverse is also true, is that if you look at something like – There are regularizations and normalization that people do, like AVE gets expanded into avenue. ST gets expanded into street. These sort of changes is what the machine augmented

review takes off the plate for the humans. This is just – There is a standard [inaudible 00:50:44] of time, like a setup and a review even if it's a [inaudible 00:50:47] and a change. This is what we try to shave off of the entire sort of pipeline.

[00:50:53] JW: Yeah, I was going to add. It was very much an iterative process where we're finding types of things that mappers were reviewing and putting them in buckets and then trying to say, "What is the low-hanging fruit of something we can just automate?" It took a long time to really find these rules, generate these categories and then ultimately like build some automated system around them.

[00:51:14] SM: I would like to give a shout out to our sort of the manual review validation lead, [inaudible 00:51:18], and we used to joke that [inaudible 00:51:21] is very good at like actually look at a UI. So this goes, this doesn't go. I was joking to Jake that [inaudible 00:51:27] stand for linearly optimized [inaudible 00:51:29] as a service.

[00:51:33] JM: The pattern of like starting with the human review process and basically studying what took the humans longer to review perhaps is a way of assessing these are the things – If I understand correctly, these are the things that we should serve to a human. The human is very fast at reviewing it. That probably means you can very easily define some rule that the human is seeing in each of these cases where the human – You start by just kicking all of these change sets to the human. The human can review some subset of them very quickly. Some subset of them fairly slowly. The ones they can review more quickly, you can easily define a rule around that and put that into your rules engine that's going to be in the automated review process. Am I understanding that correctly?

[00:52:22] SM: Yes, you got it.

[00:52:23] YW: Yeah.

[00:52:24] SM: The enemy is time, right? What you're optimizing for is, as you said, freshness and correctness. Freshness is a time [inaudible 00:52:31], and correctness is basically an accuracy metric. What we are trying to do is bring down the baseline atomic time required to review a change. Down to the fact that only if we feel that a machine cannot make a nuanced

decision here, it should go to a human. Let's not waste the manual reviewer's time. That's pretty much all we are looking for.

[00:52:58] JM: This is a pattern in defining automated review systems. I mean, these kind of automated review versus human review systems are proliferate these days when you have all these user-generated content and you have these machine learning-based like feed generation algorithms. I'm sure, Facebook has a lot of these kinds of systems built internally. What kinds of broad lessons have you learned about how to build these kinds of human review systems that are being used to kind of bootstrap the automated review systems?

[00:53:36] SM: I mean, I can speak to the other efforts that are there. Obviously, those teams, hopefully you'll get to do calls with them. For the map data, since we are actually editing public data, we are actually validating public data. It is very important for us, as Jake pointed out with the very-well publicized case of New York City changing to a racial slut. I mean, our manager, Kevin, he basically used to joke that we have a metrics of [inaudible 00:54:02] averted. Like how many severe events have we averted.

Again, I come back to Sansu which says the greatest generals are unknown [inaudible 00:54:10] to fight wars. A lot of these goes down to the fact that we stop bad things from getting in. That means our metrics – And at the same time, we want to consume a lot of the map changes that are coming down the line. It's always a tightrope walk between basically a high recall rejection versus a high precision accept. When we are in doubt about something, we should stop it. But if that doubt is blow a threshold, it's probably good to go unless it hits some other like threat markers.

That's the lesson we learned, is it's always going to be [inaudible 00:54:47]. The work is never done, because it's also an arms race, right? We have always been public about how we validate the data and all of that. The lay of the land will change. The kind of vandalism and validates that will come in will change. Our system has to adapt to this. This, as Jake said, is a constantly [inaudible 00:55:07] process and this is where the nuance of the manual review plays a lot, is we learn from them. Whenever we look at them, looking at changes, the flag rejects and sort of changes. That took them a long time and we then go back to it and see how we can map this. What part of this can be automated and what part of it has to be a human review?

[00:55:26] JM: All right. Well, we're nearing the end of our time, but I just want to pull back and get a little bit of a perspective on what you guys are working on today and the difficult problems that you've been encountering more recently. What are the prime focuses right now of trying to create this accurate map of the world?

[00:55:50] SM: If you look at how the larger team we roll into, which we call world.ai. We have this website called mapping with AI where we give tools for people to edit at leverage machine learning generated data, which is organized into three parts; read, write and display. The read part is what we are talking mostly about today, which is ingestion and integrity and making sure that the changes are good and that we produce something that can be worked on. The write team works on producing tools like a RapidUI, which actually is built on the open source ID and allows you to use this machine learning predictions for roads and buildings into like sort of onion skinning them on to the UI so that it can quickly map places and validate them.

The third part is display, which is basically our own, which displays the Facebook map. In all three of these, the problem is evolving. If you look at the number of changes that go in daily, I have seen that basically ramp up over the last two, three years they've been looking at it, is that OSM is going mainstream, which means there is a lot more data, which means there is a lot more variance and volume in the kind of bad things that we have to keep out.

As I said before, this is a moving target. The goal post is on a treadmill, or rather a conveyor belt. We have to like Wayne Gretzky our way to how this thing can be, like shoot for where the bad changes will be, and that's where a lot of our time goes in. The other part that we spend a lot of time in improving is if you look at the split between what goes for manual review and human review, there is a big part about semantics, which we have been playing with the approaches like can we create vector embedding of the kind of changes people do? Apply like financial risk modeling sort of approaches to evaluating like how risky a change is. This work is already going on.

[00:57:49] JM: Okay, guys. Well, I want to thank you both for coming on this show, it's been a real pleasure talking to you.

[00:57:53] SM: Jeff, before we go, one thing I would like to call out is Facebook's use of OpenStreetMap is not limited to our products. The organization, large organization we're part of also has an [inaudible 00:58:02] called Data for Good. Where we make a lot of visualizations and data that some of them derive from OpenStreetMaps or from other dataset is available for public download. You can just search for it and take a look at what that is.

[00:58:16] JM: Very cool. Yeah, that tool for human review, that's open source, right?

[00:58:21] SM: No. The tool for human review, because it is tied so deeply into the Facebook data systems and all of that, it's not open source. The tool for editing the map, which is Rapid, is open source.

[00:58:31] JM: All right. Okay. Yes.

[00:58:33] SM: Yeah.

[00:58:33] JM: Yeah. Okay. So I've seen some screenshots of Rapid, that's a pretty cool product, tool. Well, thank you guys. Thanks both for coming on.

[00:58:39] SM: All right. Bye, Jeff.

[00:58:40] JW: Yeah, thank you very much. This was fun.

[END OF INTERVIEW]

[00:58:50] JM: As a company grows, the software infrastructure becomes a large complex distributed system. Without standardized applications or security policies, it can become difficult to oversee all the vulnerabilities that might exist across all of your physical machines, virtual machines, containers and cloud services. ExtraHop is a cloud-native security company that detects threats across your hybrid infrastructure. ExtraHop has vulnerability detection running up and down your networking stock from L2 to L7 and it helps you spot, investigate and respond to anomalous behavior using more than 100 machine learning models.

At extrahop.com/cloud, you can learn about how ExtraHop delivers cloud-native network detection and response. ExtraHop will help you find misconfigurations and blind spots in your infrastructure and stay in compliance. Understand your identity and access management payloads to look for credential harvesting and brute force attacks and automate the security settings of your cloud provider integrations. Visit extrahop.com/cloud to find out how ExtraHop can help you secure your enterprise.

Thank you to ExtraHop for being a sponsor of Software Engineering Daily, if you want to check out ExtraHop and support the show, go to extrahop.com/cloud.

[END]