EPISODE 675

[INTRODUCTION]

[0:00:00.3] JM: Google has two consumer operating systems, Android and Chrome. The Android operating system has been widely deployed on mobile devices. Chrome is an operating system for laptops and tablets. It was originally based around the Chrome browser.

For several years, these two ecosystems were mostly separate. You could not run android apps on a Chrome operating system. Shahid Hussain and Stefan Kuhne are engineers at Google who worked on support for Android apps on Chrome OS. This was an implementation of Android on Chrome, and this implementation evolves running the Android operating system in a Linux container on the host Chrome operating system.

In today's show, Shahid and Stefan compare the Android and Chrome operating system platforms. They explain why Google has two different consumer operating systems and the advantages of allowing Android apps to deploy to Chrome. Shahid and Stefan also talk about the challenges of porting mobile applications to Chrome OS. As you can imagine, Android apps are made to run on small screens and tablets. How do you make them run on Chrome OS, which is a laptop or a desktop operating system mostly? It's also a tablet operating system. There are Chrome tablets.

This was an interesting episode. It was definitely a little bit out of my comfort zone, because I'm not intimately familiar with either Linux or Android. But Stefan and Shahid were really good at explaining stuff and I'm looking forward to doing more shows around Android.

[SPONSOR MESSAGE]

[00:01:50] JM: Data holds an incredible amount of value, but extracting value from data is difficult, especially for non-technical non-analyst users. As software builders, you have a unique opportunity to unlock the value of data to users through your product or service. Jaspersoft offers embeddable reports, dashboards and data visualizations that developers love.

Give users intuitive access to data in the ideal place for them to take action within your application. To check out Jaspersoft, go to softwareengineering daily.com/jaspersoft and find out how easy it is to embed reporting and analytics into your application.

Jaspersoft is great for admin dashboards or for helping your customers make data-driven decisions within your product, because it is not just your company that wants analytics. It's also your customers that want analytics.

Jaspersoft is made by TIBCO, the software company with two decades of experience in analytics and event processing. In a recent episode of Software Engineering Daily , we discussed the past, present and future of TIBCO as well as the development of Jaspersoft. In the meantime, check out Jaspersoft for yourself at softwareengineeringdaily.com/jaspersoft.

Thanks to Jaspersoft from being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:03:23] JM: Shahid Hussain, you are a product manager on Chrome OS at Google, and Stefan Kuhna, you are an engineer at Google. Guys, welcome to Software Engineering Daily.

[00:03:31] SH: Thanks so much for having us, Jeff.

[00:03:32] SK: Thank you.

[00:03:34] JM: Yeah. So I'm excited to talk to you about Android and Chrome and Android on Chrome. Let's start with a discussion of Chrome OS. Can you start by explaining how Chrome OS came to be? Why did the Google team create a new operating system?

[00:03:49] SH: Sure. A goal when we first built Chrome OS was ultimately to try and build a better computing experience for everyone. We started in 2010 with a device called the Cr-48. It was a device that was sort of a test device for us, and things have gone from there. With that initial kind of idea, was that we felt there was an opportunity build laptop operating system that's shaped for the way that people use computers today. So something that's fast and easy to use and secure, and that sort of Mantra is still the way that we think about Chrome OS today. Of course, things have changed a lot since we first started, but that core idea very much remains the same.

[00:04:31] JM: There's Chromium OS and Chrome OS. What are the differences between those two?

[00:04:36] SK: So the difference between them is essentially that the Chromium OS is a public domain version. So you can actually download the source code and you can actually build it yourself if you want. So it doesn't really contain any special magic Google source code in there, but it is also of course not able to do everything what we can do in Chromebooks. For example, [inaudible 00:04:55] would not be in there and the Play Store is not in there, but basically you can actually run everything. Chrome OS itself is an [inaudible 00:05:05] special thing, which runs on Chromebooks, which has all these kind of things build in.

[00:05:10] JM: Can any devices run Chrome OS? What are the device constraints for a Chrome OS?

[00:05:14] SH: I think the way that we think about is we want the hardware and the software to kind of work together as seamlessly as possible. We don't want users have to worry about things like installing drivers. So from that perspective, we work closely with people who build hardware or OEM partners to ensure that Chrome OS works really well with that specific unit. So I wouldn't necessarily think about it from the perspective of buying like a random PC hardware putting Chrome OS on it. It's more buying a Chromebook that has hardware and software specifically designed to work together. Does that make sense?

[00:05:52] JM: Yeah, it does. Now, can I have a tablet that runs Chrome OS? Can Chrome OS run on tablets?

[00:05:59] SH: Yes, in fact you can buy one, the Acer Chromebook Tab 10. That's what we call a slate, but also ship on lots of convertible PCs, like the Pixelbook or Samsung Chromebook Plus. Of course, those you can convert to a tablet form factor. They're thin and light enough to actually be decent tablets.

[00:06:19] JM: How would the tablet running Android differ from a tablet running Chrome OS?

[00:06:25] SH: Okay. Maybe I can set some context or help. We've been seeing demand for devices that startle both mobile and laptop form factors, and that's why you can see a number of Chromebooks that are convertible to tablet. But those vanilla tablets, the slates that I mentioned earlier are really just an evolution of what we've been doing already.

On the Android side, we've got lots of partners that already release great tablets, and the hardware is fairly similar, big screen and SOC that works well for mobile.

[00:06:58] JM: Google had two different operating systems being developed in parallel with Android and Chrome OS and both of those projects were going for several years. What were the different learnings that were established in those two disjoint teams?

[00:07:14] SK: Well, one operating system was of course made or meant to be done or be used by phones, and you can actually see that if you are looking through all the APIs and all the stuff what application developers have to do, you can definitely see that it was never really meant to be run on a desktop computer. So all these kind of additions which were coming into play to make this possible came in the last two API levels. So therefore, this is actually something which is really new.

Chrome OS on the other hand was really meant to be done or meant to be used for Chrome internet. So it's like a remote terminal if you want, even though this is of course totally understanding what is. But the thing is like it was more meant to be very simple. You don't have to set up anything. You simply log in and you have everything there. Which was a totally different topic like on a phone for example.

Therefore, one thing was a clamshell factor, means like you have a keyboard and something like a desktop, and the other thing is really meant to be used as a phone.

[00:08:15] JM: Without drilling into too much detail, maybe we could talk briefly about how Android apps run on Android devices. Can you describe the runtime that Android apps run on Android devices?

[00:08:30] SK: Right. Okay. In order to run that, first off, okay, I'm a little bit technical here. I hope this is fine. First off, Android is using Zygote process, which is essentially having everything what the process needs in order to be started. It has a kind of a shell. All it has to do is it has to duplicate, to copy the shell and then it can actually start its process in it.

The benefit of that is you have less resources you have to use. It's much faster of course. Which is by the way similar to what Chrome is doing with a renderer, but let's leave that all for the moment. Then once that is done, it is actually – It is then starting the program on there, which could be either native code, or it could be Java code. If it's Java code, it needs to be interpreted into the [inaudible 00:09:13] code. Then it's being started. From then on, it's actually communicating with the staff it has already inside the Zygote process with the operating system through a means of binder calls, which is some kind of API or internal API level if you want, which is underneath [inaudible 00:09:32].

[00:09:33] JM: To contrast that again with the Chrome OS model, the Chrome OS model is more your – Just having a light interface to some remote service that's sitting on a server somewhere else.

[00:09:48] SK: I wouldn't call it this way. So Chrome essentially is like it's a browser, right? So it's downloading the content of the webpage, and the webpage itself could run all native. For example, you can go into the Chrome store, you can download some 3D games, which are really downloading a whole bunch of JavaScript code including some OpenGL stuff and whatnot and it's running entirely on your machine.

Therefore, as a renderer from the Chrome OS side, is then running this JavaScript program and you can actually run it then locally entirely. So you could of course, on the other hand, do something else. You could of course say, "Hey, I'm streaming now as H.264 video," for example, "and I'm transferring all the keyboard input shortcuts to the server," and then you can actually run, for example, games like Steam [inaudible 00:10:36] box, something like that is thinkable, right?

In that way, then you have of course a pure service-based concept where you have really only a terminal at that point in time.

[00:10:47] SH: Yeah, that's right. I think just to echo on Stefan's point, there's a ton of software that runs client side on Chrome OS. It's not, from our perspective, like a thin client, or that kind of solution, and I don't think we think about it in terms of like, "Hey, we want to do one of the other." I think for us, we just want to be able to deliver like the best experience to our users. In some cases, that means that a piece of code needs to run locally on the client, whether that's via Chrome and something like JavaScript, or WebAssembly, or whether that's something like an Android app, or now Linux apps. In some cases, the best thing is for the processing to be done remotely. So it's really dependent on the application.

[00:11:25] JM: You can also imagine a sliding scale for applications. So Gmail for example, if my Gmail were to eagerly load everything in my mailbox on to my Chromebook, the performance might be a lot better than having it have to dynamically pull things overtime as I request them.

[00:11:43] SH: That's absolutely right. I think Gmail is a pretty good example of one, where there is a balance. In the latest version of Gmail, you have things like offline support, and it caches some number of emails to be able to retrieve those quickly. But rather than going the whole way and saying, "Okay, we're going to have your entire inbox available on a local machine." That just takes up too much space and it takes time for the local machine to pass through that and retrieve the relevant bit of data. That's a case where there's some functionality client side and some functionality on the server side, and that's what brings us that experience in that case.

[00:12:18] JM: When did the discussions around having Android apps run on Chrome, or perhaps there were also discussions of Chrome. Well, I guess Chrome does run on Android. How did the discussions of having the two operating systems overlap? When did those discussions begin?

[00:12:35] SK: I think that is actually a long story. The thing is like, okay, there's some history to the whole thing. Therefore I think officially we were starting with, I think, Marshmallow, to actually think about these kind of lines where you can actually see our first implementation, which came out three years ago at IO.

[00:12:52] SH: Then before that, is the ARC Original Project.

[00:12:54] SK: Exactly. Before, we had this ARC Original Project, but the thing is like that did never had the blessing in any way from Android itself. So therefore, it was like an experiment to see if we can actually use [inaudible 00:13:08] all of our infrastructure on the Chrome side to actually do an emulation of the old thing.

[00:13:14] SH: So just to sort of get us the chronology. Stefan mentioned, we started with the original ARC project, and that project was around, as Stefan mentioned, using sort of [inaudible 00:13:28] on the client side to get apps to run, but in those cases, those apps needed to be specifically built for that system. So what we couldn't do is say, "Okay, apps on the Play Store are generally available." We had to go and work specifically with those developers to have them shift into that project.

I think it's the point where IO, we had a couple of devices where we showed what was coming. At that point, we had to think [inaudible 00:13:59] one of the early Acer Chromebook Flips, the C-100, running Android on M in a container in the same way that we do it today and showed how that could run any Android app without the need for a modification. By the way, I do want to qualify that Android apps, and we can go into this in a little more detail, do benefit from some optimization work, but they don't need modification. Many apps run just fine off the bat, and that ability is what's been able to enable us to offer the whole of the Play Store on Chrome OS.

[00:14:30] SK: Also, just for the general understanding, the old ARC Project was basically you had only a single process, which is already against what Android is doing. You had to actually include part of the operating system into your application space. There was a lot of fudging and stuff needed in order to make this even work.

Because of that, there was – For example, storage was done via HTML storage, which was lousy and slow and whatnot, and applications had to be modified. I mean, it was not maintainable at all, it was created for Lollipop back then. The thing is like going to the next level to the next API level would have been a multiyear effort. So therefore, it was not maintainable at all anyways.

On top of everything, the user wouldn't have gotten ever a Play Store and would have never really – It would have been something for a [inaudible 00:15:23]. I don't know, a handful of applications, which could have run on it. But this way, by using the new method, you were able to essentially run every Android application on there.

[00:15:34] JM: If I understand correctly, you're saying A-R-C, ARC. ARC was a prototype runtime for Android on Chrome and in contrast with today's current container model for Android.

[00:15:46] SK: It was an attempt to get Android applications on to Chrome. Let's take it this way. It was an emulated version of Android, if you want, and it had a lot of drawbacks.

[00:15:57] JM: What were the drawbacks?

[00:15:58] SK: Oh, it's slower. Application developers had to do – They had to do something to their application to even run on there. There were a lot of limitations. Like for example, on the Google server, there was only a little bit of that stuff there, then the whole operating system was running in the same process and so on and so on. Which means like for application's perspective, it wasn't entirely different system and it would have actually added to the fragmentation on Android.

[SPONSOR MESSAGE]

[00:16:36] JM: Your audience is most likely global. Your customers are everywhere. They're in different countries speaking different languages. For your product or service to reach these new markets, you'll need a reliable solution to localize your digital content quickly. Transifex is a SaaS based localization and translation platform that easily integrates with your Agile development process.

Your software, your websites, your games, apps, video subtitles and more can all be translated with Transifex. You can use Transifex with in-house translation teams, language service providers. You can even crowd source your translations. If you're a developer who is ready to reach a global audience, check out Transifex. You can visit transifex.com/sedaily and sign up for a free 15-day trial.

With Transifex, source content and translations are automatically synced to a global content repository that's accessible at any time. Translators work on live content within the development cycle, eliminating the need for freezes or batched translations. Whether you are translating a website, a game, a mobile app or even video subtitles, Transifex gives developers the powerful tools needed to manage the software localization process.

Sign up for a free 15 day trial and support Software Engineering Daily by going to transifex.com/sedaily. That's transifex.com/sedaily.

[INTERVIEW CONTINUED]

[00:18:24] JM: How did Google Play Services end up getting exposed in Chrome OS? As I understand, Google Play Services it it’s a large component of the Android runtime, which is a significant component. You have your Android apps running and they can share these different Google Play Services that might help out your Gmail app, and your Google Maps app, but all these different things. Porting that to Chrome. If you could talk a bit about Android, or Google Play Services.

[00:18:51] SK: Right. So the thing is – So the big difference which you have to understand here is that having a container means basically that you’re running the full Android operating system like it is on a phone on your machine side-by-side with Chrome. Which means like the full feature set, whatever Android has is actually running on the side. Therefore, since we are actually just – Well, we are justifying all the [inaudible 00:19:17] means that we are passing CTS tests, we are officially an Android client and everything works. Because of that, we are actually getting the stamp that we can actually use the Play Store. The Play Store is now really running on Android or the same machine. So that is the big difference.

[00:19:36] JM: Got it. Can you describe that current Android container model in more detail? We've done a bunch of shows recently on Docker and Kubernetes and so on, but maybe you could talk more about the Android container model for Chrome?

[00:19:50] SK: Okay. So there we have, first, SE Linux, which is essentially giving us the base from the Linux kernel, and we have, of course – With this, we have then two different container which have then their own access rights and whatnot. Android is running now in its own container and it's communicating over some channels with Chrome, like there is a valiant client, which is implemented on the Android side, which is now communicating with the Valiant client on the Chrome side, and Chrome has now the compositor. That is the same, which is actually giving you the desktop in the end, because it needs to actually know all the windows on Chrome as well as the one from Android. Therefore, it will actually get in all the different layers, all the GPU content and it will actually map everything together under the same desktop. That is done through the valiant layer.

Then we have some additional things. I mean, perhaps all of the stuff what Android is doing is directly through the kernel and memory allocation and whatnot. Even the network is also then abstracted through there. But then there are some other channels for general configuration and what not, which is going over an internal module interface, which is taking care of other things, like login and whatnot.

[00:21:03] JM: Some people might be thinking – So in this model, I've got an operating system and then I've got another operating system running in a container on the same machine. How on earth am I going to have enough resources to handle two operating systems at once? How you would address those questions?

[00:21:19] SK: Yeah. So that is actually an excellent question, especially if you think about the fact that we have also another container, which is Linux, which you can also run on the side. Therefore, each of these things will actually use their own memory and their own CPU resources and whatnot.

Well, the answer is very simple. Usually, you are running only one application at a time. I mean, it's not really – You are using multitasking, sure, of course, but you are always using only the focused application really at that point in time.

Android has a lot of power management features build in, like for example the run state management, like for example, an application is either running, or stopped, or destroyed. As soon as an application becomes invisible, like for example you're minimizing it or another application becomes full screen, then it will become stopped and then it will be destroyed at that point in time sooner or later. Therefore, in a way you are actually getting rid of all the unnecessary [inaudible 00:22:17], which is actually round.

Another thing to look at it is like if you're looking to the memory consumption of certain apps, I mean, Chrome is – Depending on what page you're on, is of course also consuming a lot of power, means like memory and whatnot. Therefore, it's not like that.

Whereas Android applications, they're usually relatively small. Therefore, it's a balance between those two. Then Linux comes on top of everything. But they have to behave, and usually you don't really have tens of applications turned on at the same time. I think there are some studies that usually for the most parts they are three or – I think two to three windows open at the time at all times. Therefore, you don't really have that many things running at the same time. If you're a specialist, I mean, your screen real estate is pretty limited. Therefore, you cannot really have everything side-by-side to organize your windows, to have 20 windows at the same time open. It will be pretty much impossible, I think.
Therefore, as soon as the window is covered, it's being thrown out on memory and it doesn't get CPU power anymore, unless it is a background service in Android.

[00:23:32] JM: Is it important to be able to be allow these different operating systems that are running on the same machine between Linux, Android and Chrome to be able to share resources? Or to what degree do you need to have sharing, or is it fairly well bounded between them?

[00:23:49] SK: Another excellent question. So the thing is sharing off resources is – So the GPU resources are in a way shared if you want, because the thing is you need to share them, otherwise the compositor on Chrome would not be able to show everything composited on to one screen. Therefore, you need to definitely share the GPU resources in a way. At least the flat memory so that the GPU can pick it up and draw something with it.

So everything else, usually we are trying to decouple these tools as much as possible, because if one thing gets breached, you don't really want to have your entire system going because of security problems. Therefore, there is of course – There are some barriers in place. For example, Android can only see parts of the file system. It cannot see everything. Therefore, there are some limitations in regards of what each of these things can see.

If a user for example is dragging and dropping something from one window into another and one window might be, for example, Linux, and the other one might be, for example, Android, then there is communication channel between these operating systems to make this data change available.

[00:24:57] JM: When I think about the applications that I use on Android, they are typically made for the window size of a smartphone. They're made for the form factor of a smartphone. They're made for the touch gesture type interactions of a smartphone, or perhaps for a tablet device if there's also a tablet version. Well, I guess even if you just have a phone version, it can also run on a tablet. But in any case, it's not the same as I think most people who use Chrome OS use a laptop version. So what's the model for getting the handheld tablet or smartphone application medium ported to the laptop medium for Chrome OS users?

[00:25:38] SH: I think that is – We've been working with developers over the last year or so to try and identify issues that are cropping up and find ways to solve them and be very clear in our documentation and our feedback about how to do that. So what we've found is that there are really four things that crop up when we try and run Android apps on Chrome OS.

The first is that the screens are a ton wider. As you mentioned, a typical – In fact, in Chrome OS, the smallest device we have is 10.1 inches. The largest we have is 15-inch screen. Whereas your mobile screens are typically between 4 and 6 and a bit inches. So that's quite different. So some of the things we see there are user interface elements, which could be sort of panned left inside some other container element in the UI. Then you have sort of on a wide, this massive white space inadvertently appearing.

The second thing that we see is that the laptops are default landscape. So when you open up a laptop device in its kind of laptop configuration, the clamshell configuration. So it's not sort of converted into a tablet. Then by default your screen is going to be landscaped. Some Android apps either focus primarily on or are only available in portrait. So that's one of the other things that an app developer will need to keep in mind when building the app.

A couple of other things that are worth mentioning, the first is maybe an obvious one. As he mentioned, the input is different. Almost sort of non-Chrome OS Android devices have touch screens. So as a consequence, you see things in the user interface, like there are touch targets that are finger sized rather than like a slider control that is made for a mouse. So that is something that we need to think about when we're building an interface for an app is this thing that can adapt in both of these circumstances. Many of our Chrome OS devices have touch screens, but not all of them.

So one of our more popular devices, the Samsung Chromebook 3, which is kind of super affordable, available at Best Buy by the way. If you want to try that, that is not going to touch screen, but it does run Android apps. So in that case, the app needs to be able to understand things like multi touch. Does the app rely on that kind of interface, like a pinch and zoom in order to play the game or look at something more closely? If it does, then there needs to be an alternative. Then the last one is a windowing system.

So one of the things that we find is that many app developers will test understandably first on a mobile device, and most apps on a mobile device don't go through window resizing. You can't actually resize an app on like a version of Android by doing split screen and then moving the split around. But it's less obvious and way less common than it could be on a multi-window system.

So, really, our goal has been to educate developers on these kind of issues and essentially like encourage people to test their apps when they're building them on a Chrome OS device, because now if you build an app and you release it on the place or by default, that is available on Chrome OS. So what we don't want is app developers to hear from Chromebook users saying like, "Hey, we love your app, it's great, but it doesn't really work that well on Chrome OS." We want to head that off by asking developers to pick up a Chromebook and do some testing. We also ship an emulator that works inside Android Studio. You can do it that way too. But however they do it, check in and ensure that these issues aren't going to occur. Many which are super obvious as soon as you load up the app into a Chromebook.

[00:29:21] JM: To take a step back, for a long time, people have simply developed a web version of their mobile app, or perhaps a mobile version of their web app depending on how you look at it. Is the main gain here of putting Android apps on Chrome OS, is the main game that you're envisioning a world where no longer do you have to write an app for multiple surfaces?

[00:29:49] SH: So I'd maybe think about this from a user's perspective. So ultimately, we just want users to sit down on a Chromebook and run the app that they want to run. Some of the apps, they want to run around the web and some are on Android. Bringing the Android framework to Chrome OS has enabled the Android part of that.

So if a user really wants to run Roblox, and Android game, they can now do that. They don't and shouldn't have to care where it comes from. They just want to play. Because you mentioned web in your question, and Google works hard on improving the web too. The Chrome and Chrome OS teams are actually part of the same team here. There's a really close partnership.

I just want to note though, I'd maybe challenge or note that we're putting mobile apps on Chrome OS. We're providing the Android framework and developers can use it in whatever way they want to. For example, we've worked with the team at Squid to ensure their app, a note taking app. Works great on Chrome OS. They've told us they're making 21% of their revenue now on Chromebooks. So is that a mobile app anymore? Maybe not. It's just an Android app that works across multiple surfaces. That's something that web apps and Android apps can do that too.

[00:31:05] SK: There is of course a thing that there are two million applications already out there for Android, which we can simply dive into. Viola! There they are. Everything is running on one system.

[00:31:15] JM: That's a good point. Yeah, to the point of flexibility, I think a developer who wants to write an app to target both Android and Chrome, now they could either write it for Android and have it run on Chrome, or they could also, overtime, as you mentioned with WebAssembly, with the web just getting better, perhaps they could write a progressive web app and just have a progressive web app that runs really well on Android and would also run of course well on Chrome, because it's a web app.

[00:31:42] SH: Yeah, that's exactly right. I think our approach to this is really to give the developers as much flexibility as possible and do the best that we can in building the tools and the infrastructure and the platform regardless of whatever choice they want to make.

[00:31:56] JM: I would like to dive back into some of the engineering challenges of this project. What about permissions and security issues? Sharing these kinds of things between the different runtimes and making sure, for example, if I'm logged in to something, if I log I give credentials to something on Chrome OS, the ideal world is my Android container is aware of those permissions and can recognize that I'm logged in and I don't have to log in once again. Was that a challenge to implement?

[00:32:30] SK: Actually, that comes pretty much naturally I would say, because the thing is on Chrome OS, you have to log in in order to have a session running at the first place. Then the thing is on a phone, in order to set it up, you have to actually, well, log in into a user account. The thing is like once you have figured out that both are essentially the same accounts, you can actually set up your Android space once with your credentials from a user. Then you have the credentials from Chrome on the other side. Therefore, I think I would say this particular thing comes naturally.

[00:33:07] JM: Tell me more about window management, because that sounds like it was a really difficult challenge in getting Android working on Chrome OS.

[00:33:13] SK: So there was one problem with cameras. So if you look, for example, at a camera, and you look at a phone, you figure out that the camera is always the same orientation as your phone, right? There is a specification for all Android, which is essentially dictating that these orientations are the same. Now, if you have an application running, and the application wants to be portrait only, right? But your screen is landscape. The camera itself is now of course also landscape, because it's following what Android does. But a clamshell device means a notebook device, well, you cannot really have that portrait at that point in time. It's landscape, right? But you have to show now this portrait application. What do you do now as a video coming from the camera?

So now you have to understand that Android itself, they have of course – Since this is a basic requirement from the operating system, they were nowhere adding kind of API calls, which allows you to rotate the picture at this point in time by 90 degrees, because by definition, the video was always the same orientation as your phone.

Let's assume for a moment you have fixed this now for a particular application and for this particular case. What happens now is the application suddenly changes its mind and says, "Oh, I want to be now a landscape." It does something really wonky there. Then suddenly the window is suddenly rotated by 90 degrees, or the person is stretched then suddenly, or slipped down or something. This is something like where the original Android definition had of course no notebook form factor in mind. As the definition of this API is already out since ever, we cannot really change that, at least not overnight.

So this is something where applications developers, they have to help us doing the right thing, and we cannot really do much for them. We were trying, and it was really – It's a nearly unsolvable problem at this point in time. We can do our best to try to actually fix everything, but it will not be good enough if the application is doing something funny.

[00:35:27] JM: Okay. Can you start by describing how the Chrome OS project came to be and why Google decided to invest in a new operating system?

[00:35:35] SH: Sure. Our goal with Chrome OS is to build a better computing for everyone, and we have first started 8 years ago with the Cr-48 and we've worked with many of our partners to launch many devices since then. We thought that was an opportunity to build a laptop operating system that's shaped for the way that people use computers today and a system that's fast, that's easy to use, that's secure. That's very much still a mantra.

[00:36:05] JM: What are the most appealing features of Chrome OS?

[00:36:08] SH: So maybe let me explain this one another way. So let's say you've come back from IKEA with a brand new HAMNES and you want to put it together.

[00:36:18] JM: Sorry. A brand new what?

[00:36:19] SH: HAMNES, it's a book case.

[00:36:22] JM: Okay. Got it.

[00:36:23] SH: So let's say you want to put your bookcase together. I guess I assume that people go to IKEA as much as I do, but maybe that's not the case.

[00:36:30] JM: I just don't know the names. I know I'm sitting on an IDEA desk right now.

[00:36:33] SH: Oh, all right. All right. Great. I'm a fan. So let's say you want to put together your bookcase. The last thing you want to do is figure out how to use your screwdriver. Your screwdriver is the tool that you use to get something done that you actually want, and we feel the same way about Chrome OS. Most people don't sit down at their computer they want to deal with their operating system. They want to do something. They want to write an email, Facebook, whatever. The operating system is the tool that they use to do that.

So I think Chrome OS's best feature is actually what we don't do. We don't nag you to update your antivirus definitions. We don't make you sit around the 20 minutes while your computer updates, and we think users shouldn't have to do those things in 2018.

I think that's how we want people to use our computers today without any of those hassles, and that's really our goal. There's a ton of features that we can go through, but I think, for me, that's probably the biggest one.

[00:37:31] SK: I think, for me, actually it's more like you get a new machine, and I constantly get new machines. I log in and everything is there exactly as on my other machine. Simply, that's totally awesome. I mean, doing the same thing on other systems without naming names takes a while. You're sitting there, stretching your head, "Why is now this option like that? Where's that thing again?" Here, you don't have to do anything. Everything is simply exactly as you have seen it on the other system.

[SPONSOR MESSAGE]

[00:38:08] JM: Stop wasting engineering time and cycles on fixing security holes way too late in the software development lifecycle. Start with a secure foundation before coding starts. ActiveState gives your engineers a way to bake security into your language's runtime. Ensure security and compliance at runtime.

A snapshot of information about your application is sent to the active state platform. Package names, aversions and licenses and the snapshot is sent each time the application is run or a new package is loaded so that you identify security vulnerabilities and out of date packages and restrictive licenses such as the GPL or the LPGL license and you identify those things before it becomes a problem.

You can get more information at activestate.com/sedaily. You want to make sure that your application is secure and compliant, and ActiveState goes a long way at helping prevent those kinds of troublesome issues from emerging too late in the software development process. So check it out at activestate.com/sedaily if you think you might be having issues with security or compliance.

Thank you to ActiveState for being a new sponsor of Software Engineering Daily.

[INTERVIEW CONTINED]

[00:39:38] JM: What were some of the biggest development challenges to getting this working?

[00:39:45] SK: You mean the security thing?

[00:39:46] JM: I mean – Sorry, more abstractly. Getting Android running on Chrome. Just getting to the abstract level.

[00:39:53] SK: Yeah. So there are many different parts. For my part, I was essentially working on window management. Window management has a lot of challenges, because as you can see, you have to actually combine everything on one desktop. This starts a question at one point on what is a window? I mean, it sounds very trial, right?

But the thing is as I mentioned earlier already, Android was never really managed to be run as a window in this pod. So if you look, for example, into what a dialogue box is on Android, essentially, it's like window – No. It's a surface where you are only using a certain portion inside. So now, how does this all translate into a window? Well, it's like you have a task, that is your window, then every activity is essentially like adding another layer, it means like, and all those 3D texts showed on top of that.

The biggest problem here is, of course, that you need to actually now let Android applications render their stuff on the Android stuff using OpenGL and everything, and then you have these surfaces and you need to transfer them over to Chrome, and Chrome needs to be able to composite everything so that it makes sense to the user what he sees. You can of course do that simply, but the thing is like then [inaudible 00:41:10] peformant and you will actually see lots of performant degradation. So there is number one.

Then Android's whole framework was not really meant to be running side-by-side with something else. So this needed to be integrated, means like the – Like, for example, looking into the input stack. The input stack was meant by Android to be secure. They're directly trying to address the kernel, trying to get all the input events, making it streamlined and fast.

[inaudible 00:41:41] putting in some kind of security measures in place so that this cannot be modified by applications, which makes of course a lot of sense. However, for us, we can of course not do that, because if the user is pressing, for example, ctrl+W on a Chromebook, he expects that the window is being closed. The thing is like if you would now give the input event not through Chrome, we would never have the chance to actually use or consume this event.

Sticking on that one, so Android applications, they of course never really thought that an input event would ever come back to someone else. There are parts of the system UI library, for example, which are simply swallowing keystroke when they think they don't use them or they don't need them.

With that, there were no hard keys working on Chrome OS. So we had to actually change, for example, we have to correct some box on the Android side to make all these kind of things work and so on and so on. There's a long tale of things which had to be fixed and made to work.

Then there are of course applications, which is I think that is the biggest problem of all, because application developers, they are fixing everything, what they see, what they can do and they are of course thinking along these kinds of lines. So if you look, for example, app on the web, "Hey, how can I figure out what's the size of my screen is?" There's an article on stack overflow and it gives you something like 32 different ways of figuring that out. Guess what? Every of these options is up-voted.

Which means like, "Oh! Look at that. There are 28 API levels and there are 32 different ways of figuring out what your screen size is." I think you get the picture. You know what? Most of these things, what is being explained there, is wrong. It's plainly wrong and you should never do that. The thing is it's out there and, yeah, someone should really clean that up.

Now, depending on which kind of call you're using, you might actually get into trouble. There are applications, I don't know say names, they are being started for the first time. They look at the screen resolution, what I think is a screen resolution, they pre-scale all their 2D assets and then they store that in their own data store and they never look at the screen resolution again.

If we change now the size of the window, yeah, guess what happens? The thing is like when you see an application failing like this, the first thing which comes up is of course, "Hey, you know what? Google lost it. They have actually a crappy software. It doesn't work." Therefore we are of course looking into these kind of cases, and most of the time it's like, "Hey, you know what? This is an application which is doing something wonky." Then we have to actually ping the application developer and let them know what it is.

But the thing is like there is, of course, yeah, two million applications, which is good? But there are two million applications and they might actually not be written right. Oh, that's actually bad. Therefore, there's a lot of work to be done and it's definitely getting better, because application developers are getting more aware. Thanks to our presentations and conversations like this. But the thing is like this is actually a very difficult thing, right? Then there are of course securities saying [inaudible 00:44:49] and whatnot. It's a long tale of things.

[00:44:53] JM: What's the process of figuring out the things on the long tale? Do you just start flipping through apps and testing them one by one?

[00:45:01] SK: No.

[00:45:03] SH: It would take a long time.

[00:45:04] SK: That would take a very long time. I mean, one thing which definitely helps is the RCT tests. I don't know how much you know about certification of Android. In order to be able to ship an Android device, you have to pass a certain amount of tests. Every time when you try to ship a new version, you have to re-pass these tests. These tests, they are running for three or four days depending on what it is.

It takes a long time to run them, and you have to pass every test. The first thing in order to be able to ship something is pass all these tests while having your functionality in place. By simply doing that, you see already a lot of things which are different. With every letter change of Android going, for example, from [inaudible 00:45:51] or something, there are new tests for new things which are being added, which may make sense in our case, or may not make sense.

For example, the lock screen. So does a lock screen make sense on Chrome OS? Not really so much, because we have our own lock screen. So we don't really care about that one. But now, if you look into the internals to figure out that a lock screen is actually using the power manager. So you need to have the power manager in place in order to even be able to ship anything. So even though we are not using the power management either because it is mostly done by

Chrome OS. Now, resolving all these conflicts and trying to make that in a way so that we can ship that and we are getting the certified thing is a hard problem.

[00:46:37] JM: Shahid, from the product management point of view, what kind of challenges have you encountered?

[00:46:44] SH: So there's a few product managers working on Android on Chrome OS in one way or another. It's a really joint effort between us, the Android team, developer relations team, our BD teams, our marketing teams, and we all have this belief in the value of bringing Android apps on to Chrome OS.

We're also super lucky to have an amazing engineering team on Chrome OS. One of whom is sat next to me. So we've been working really closely with our counterparts when it comes to prioritizing the work, launching, and the process of improving technology to the point where people are having a really positive experience with it. We've heard lots of great stories of that.

On the PM side, we've also been able to be a bridge between the core product development and app developers who are partnering with us and doing the work of optimizing their apps for Chromebooks, because ultimately, that's what this is all about, getting our users a great experience when they use apps.

[00:47:43] JM: So from what you said a little bit earlier, it sounds like the vision is for Android to continue to be this wide open operating system that deploys to lots of different surfaces, whereas Chrome OS is more going to be targeted at situations where you can optimize the hardware for dealing with Chrome OS. Are these going to remain separate projects that are going to thrive on their own, or do you envision a world where these two projects merge into a single operating system?

[00:48:19] SH: So there is no plan to merge Android and Chrome OS. We've all read a bunch of press on this topic. Some of it has been super interesting, and we totally understand the question, of course, the reason for the question. That's said, a couple of notes. Firstly, both teams work together very closely, and that's been really helpful in both directions.

For our side, we wouldn't have been able to bring Android apps to Chrome OS without a really close partnership with the Android team. For those of your listeners who have been using or trying the latest beta or dev channels of Chrome OS, especially on a Chrome tablet, or a convertible [inaudible 00:48:58] form, you'll see some of the ways that we're aligning visually as well as under the hood. Again, I want to be clear to your listeners, no plan to merge.

[00:49:08] JM: What kinds of integrations or interactions between my mobile device and my laptop do you guys anticipate developing overtime? Whether it's machine learning, or just the fact that maybe these two things play more nicely together in the future?

[00:49:26] SH: So what I might ask is stay tuned on that front. But what I would say is like let me give you a quick example. So when you're inside Google Maps on a Chrome OS device, maps is smart enough to know that it's also installed on your phone. So when you go to maps on your Chromebook and you say, "Okay, tell me where I am right now." IT will pick up your current location not from GPS on the Chromebook, because Chromebooks, currently we don't have a device out there that has an integrated GPS in.

But instead it's like, "Oh, well, this user is logged in to maps on a phone that does have GPS in it. So I can figure out where I am from that." There's this sharing of hardware across these form factors that we can benefit from because we have this app deployed in both these spaces in that case, but we're always working and thinking of ways that we can integrate from one OS to the other given that the user is logged in to both places.

So I'd say I want come on that. We're definitely working hard on thinking of other ways that we can sort of get a better experience to our users because of that.

[00:50:33] JM: Just to wrap up, we did a couple of shows about Flutter a while ago. We talked to Eric from the Flutter team, which is a new way of building mobile UIs, but it sounds like more fundamentally, it's a new way of painting user interfaces just in a more flexible manner. Have you either guys spent any time communicating with the Flutter team or do you have any plans to work with Flutter directly?

[00:50:57] SH: So we love Eric. I've met with Flutter a couple of times on this topic. Yes, absolutely. So currently, Flutter can output to a number of different places. In our case, of course you can run whatever you want on the web, and now whatever you want on Android and Linux coming very soon. So we're in this kind of position where we actually support multiple outputs from Flutter.

I think from our perspective, we love the fact that Flutter is going to help accelerate app development, and the app developer makes the choice as to where they want to output that dependent on what type of application it is, and we want to support delivering that to the user, whatever it is.

[00:51:36] JM: Okay. Sorry. Just one last question just to wrap app. WebAssembly is another topic we've touched on a lot, and you alluded a little bit to it earlier. But given that Chrome OS leverages the browser and the JavaScript engine so heavily I'd love to know what your thoughts are on WebAssembly and what application potential it's going to unlock.

[00:51:57] SH: So we're a huge fans of WebAssembly. As I mentioned earlier, the Chrome team and the Chrome OS team is the same team. So I think you had Thomas on the show a few weeks. I listened to that episode and super interesting stuff. If you build an app using WebAssembly, it will run great on a Chrome OS device. For our perspective, that's one of the reasons why we're huge fans of it. We are excited to see more and more developers use it to deploy their applications across multiple spaces and we're one of them. We continue be huge fans of the project and we're excited to see what developers do with it.

[00:52:32] JM: Okay. Well, Shahid and Stefan, I want to thank you guys for making the time to come on the show. It's been really interesting talking to you both.

[00:52:39] SH: Thank you so much for the opportunity to chat through this stuff. We really appreciate it.

[END OF INTERVIEW]

[00:52:46] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes. You can quickly provision clusters to be up and running in no time while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked into any one vendor or resource. You can continue to work with the tools that you already know, such as Helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications offline. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes. That e-book is available at aka.ms/sedaily.

[END]