EPISODE 589

[INTRODUCTION]

[0:00:00.3] JM: Voice interfaces are a newer form of communicating with computers. Alexa is a voice interface platform from Amazon. Alexa powers the Amazon Echo, as well as Alexa-enabled cars, refrigerators and dishwashers. Any developer can build a device with a voice interface using a Raspberry Pi.

Paul Cutsinger works on Echo and Alexa at Amazon. He's focused on growing the market of developers who are building for voice interfaces. In this episode, Paul describes how to design and implement a voice application for the Amazon Alexa platform. The market for voice-powered apps is so new and there has yet to be a killer app. If you like to tinker on new platforms, you will like this episode. I was surprised by how easy it sounds to build a voice app.

Personally, I use voice interfaces all the time to set timers, to find out how to tell if a cucumber has gone bad, to ask what temperature to cook a potato at, sometimes when I'm lying in bed trying to get to sleep, I will ask my nearest device to read me a Wikipedia article in the dark. These are great use cases, but I'm sure we will see something much more groundbreaking in the future. If you're interested in building something like that, then I hope this episode is of value to you.

[SPONSOR MESSAGE]

[0:01:29.9] JM: We are running an experiment to find out if Software Engineering Daily listeners are above average engineers. At triplebyte.com/sedaily, you can take a quiz to help us gather data. I took the quiz and it covered a wide range of topics; general programming ability, a little security, a little system design. It was a nice short test to measure how my practical engineering skills have changed since I started this podcast.

I will admit that, though I've gotten better at talking about software engineering, I have definitely gotten worse at actually writing code and doing software engineering myself. If you want to take that quiz yourself, you can help us gather data and take that quiz at triplybyte.com/sedaily.

We have been running this experiment for a few weeks and I'm happy to report that Software Engineering Daily listeners are absolutely crushing it so far. Triplebyte has told me that everyone who has taken the test on average is three times more likely to be in their top bracket of quiz scores.

If you're looking for a job, Triplebyte is a great place to start your search, it fast-tracks you at hundreds of top tech companies. Triplebyte takes engineers seriously and does not waste their time, which is what I try to do with Software Engineering Daily myself. I recommend checking out triplebyte.com/sedaily. That's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte, byte as in 8-bytes.

Thanks to Triplebyte for being a sponsor of Software Engineering Daily. We appreciate it.

[INTERVIEW]

[0:03:27.9] JM: Paul Cutsinger works on Echo and Alexa at Amazon. Paul, welcome to Software Engineering Daily.

[0:03:32.8] PC: Thank you for having me. This is great.

[0:03:34.6] JM: You've been in Amazon for five years. How did you find yourself on the Alexa team?

[0:03:39.4] PC: I was working on the app store team doing mobile development and really focused in on user experience and helping people make great games in particular at the time. Got enticed by the lure of voice and thinking about what it'll be like to think about the future of how people interact with their computing all around them, in their spaces and how to design that was very cool.

[0:04:02.7] JM: Yeah, and I have a good idea of how a typical software team works. I know how I would structure a team to build accounting software, or a new site, these well-defined software applications that we know how to build at this point. A voice interface, that is quite a greenfield

software project. How is this the project managed, because this idea of building a voice interface seems like a very new product to build?

[0:04:29.5] PC: It is. There's a few ways to think about this. If you want to think traditional development house, what you really have is somebody that's building your web services, because at the end of the day behind the scenes is a bunch of web services, right? You get a call in from your UI, the voice in this case, and then you handle that in your web service.

You can build it with, we use AWS Lambda and I often use node, but you could use Java, Python, whatever, any back-end, any HTTPS endpoint that you want. It's really a voice UI model and then a web service model. The design side of the house is where things change up a bit, because the differences between voice and web design, or graphical design.

[0:05:08.9] JM: How does building a voice application differ from other application development paradigms?

[0:05:15.3] PC: Well, the human parts of it are very much the same. You have a lot of empathy for your customer and how they're going to work, and that is all the same. The big difference is if you're thinking about building a website, or a mobile app, what you're really trying to do is figure out how to get everybody to use your experience, right? You're effectively telling the customer what to do through your UI and you're doing that in a way that's very easy for them to learn.

You think about the layouts, the information architecture, the design, the graphical design and all that stuff is meant to make it very learnable, but ultimately people are going to have to learn what you are putting in front of them, and that's the way they have to do it. With voice, it's different in the sense that people might say all sorts of things and what you have to do as a designer is adapt to what the people are doing.

Let me give you a specific example just to just to paint the picture more clearly. In a website, or a mobile app you might have a screen full of stuff and then an okay button at the bottom. This is up to the user to figure out, "Oh, okay. Okay is the closest fit to what I want to do in this situation," so I'm going to click the okay button, right? In voice, instead of that, the software has

to figure out what the user meant. They could say okay and that's great, but they could also say all right, next, carry on, let's keep going.

The okay function underlines that button with the label okay on it, or the utterance or the things that people say could be the same, but the way they approach your experience could be quite a bit different. Part of your job as a designer is to figure out how to reach those customers as easily and as appropriately as possible.

[0:06:54.3] JM: What are the applications, or the application types that people are using the most frequently from the variety of things that people have built?

[0:07:05.1] PC: There are a few different kinds that people are building today. I would classify them in maybe three, or four different buckets. One is a brand who is trying to get some awareness about their experience, and get it out to voice where people are. That's a great example that would be like tide. If you want get a stain out of something, you just ask tide and they'll walk you through it.

Another one is more of a transactional an experience, where maybe you want to order pizza, or you want to order your Uber, that kind of thing. Then there's a lot of games. There's games we can do adventures and interactive fiction stuff, or there's like Yes Sire, this game where you can try to grow your kingdom and make decisions back and forth. If you step back from it all, the ones that work best are the ones that realize the moment that you have with the customers when you're further than an arm's length away, you're across the room and want to do something.

The super, super, super simple version of this is set a timer. I would have never said to you, “Wow, setting a timer is super hard.” I could just walk up to my microwave and hit the three button, right? Now, I can be across the room and just say set a timer for three minutes and it sets it. That notion of your technology is always within the sphere of how long your arms are, that's where how far your phone is, or your laptop, or whatever. Now you can go all the way across the room.

Experiences that allow you to really enjoy that help a lot. Another example would be what else? A game heads-up would be a good one, where instead of me playing it by myself on my phone, like staring at that, the whole room is playing it together. It's more of a communal experience, so those are pretty good examples of different kinds of experiences people are building now.

[0:08:43.2] JM: Indeed. I found setting a timer to be the killer app strangely enough.

[0:08:48.2] PC: It's weird, right?

[0:08:49.3] JM: Setting a timer, knowing what temperature to cook a certain type of meal at, those kinds of things are quite killer. I do wonder, do you think every application should have some kind of voice functionality built in?

[0:09:05.1] PC: Well, I think that depends upon the scenario. I see a lot of apps, like mobile apps that will have a voice keyboard and maybe that's useful, maybe it's not depending upon the situation, I guess. By a voice keyboard, I mean, that you can just say stuff and as if I were typing on my keyboard, it's like a shortcut for keyboards. I think that's okay, but that's not the experience that it brings you the magic that you were just describing with the simple timer, or the turn on the lights. It's really that far field, long distance experience, or a more conversational open-ended experience. One where I don't have to learn and conform to the design you've built.

I guess, I think of it this way, every app has an underlying customer need and those customer needs are real and genuine and those customers might want to interact with their computing in a bunch of different ways. Maybe it's with a desktop, or mobile phone or whatever. I don't necessarily believe that every app as it stands should just be moved over to voice as it is, right? You should think about the underlying scenarios behind it and bring out what is the essence of that. What makes it better in voice, and think about that.

[0:10:07.8] JM: One thing I wonder is how early it is. Obviously, there are some super useful applications for voice today. We can zoom forward 10 years into the future and we know that voice is going to be massively important. It's no surprise that Amazon is investing in it a lot, but the roadmap in between now and 10 years from now, there's a few things that I wonder how important they will be.

One is obviously the classes, like when do you have classes, or the contact lens that have a computer in them, so you have a visual interface to accompany, the voice interface all the time, a little bit lower-hanging fruit would be just having voice devices around everywhere in your house. When your microwave has a voice interface, when your refrigerator has a voice interface, when you are never afraid of it being present. Then there's also when businesses will be adopting these things on a regular basis and you can reliably have a voice interface when you walk into a waiting room at a doctor's office, for example, or when the doctor is examining you and they always have a voice interface available.

There's also how quickly will people adopt Bluetooth ear device. We're just talking before the call, your new computer does not have a way to plug in headphones. The same thing happened with my smart phone. We are absolutely being shuttled into having Bluetooth devices that are going to be on our heads more often, so it'll be easier to talk to them. What I wonder is how important is it to get to some of these milestones before we start to see really, really important applications be built, or are we going to see important applications be built in the current modality of voice interaction?

[0:11:52.8] PC: I think it's both. You could also make the argument that we've come a long way to get where we are. The confluence of far-field microphone arrays, natural language understanding, the different levels of AI that are at play now to really increase the accuracy, like all those things have converged. They've been in the works for a long time.

I think we're at that stage where we've got that this-feels-cool moment, and you can see the potential. The really, really exciting experiences are even more yet to come. If I compare it back to, I don’t know, it seems like every big technology revolution. I mean, I remember teaching people how to use a mouse. I play solitaire, so you learn how to double click and click and drag, these are new gestures, right? With mobile, if you remember that first moment where you probably got the Internet in your pocket and you're like, “I have the Internet everywhere I go. This is so amazing.” At the same time, you were doing a lot of pinching and zooming, because website developers just didn't understand what it meant to have a responsive design, or the fact that everybody has a seven millimeter tap target on the end of their finger, right?

That first part was great and muse and beautiful magical, but it also had a lot of growth. It took companies that would really dig in and understand what it meant to be direct manipulation and just work with different gestures and like, who knew that there's games where you touch and drag to pull the power and then letting go of the screen is the shoot button. That wasn't a concept in web, or mobile, or web before mobile.

I think that's where we are now, is where we have a lot of very obvious straightforward things like call and response style stuff; set a timer, turn on the light. Then we're starting to get into little deeper stuff with playing the games and ordering pizzas and those kinds of things. I think the time is ripe for some indie developer out there to go, what is voice really all about and push us into this next wave. That'll drive all those other innovations you were describing too, right? Sort of this cyclonic effect I guess, where we all get together and start piling on.

[SPONSOR MESSAGE]

[0:14:03.1] JM: The octopus, a sea creature known for its intelligence and flexibility. Octopus Deploy, a friendly deployment automation tool for deploying applications like .NET apps, Java apps and more. Ask any developer and they'll tell you that it's never fun pushing code at 5 p.m. on a Friday and then crossing your fingers hoping for the best. We've all been there. We've all done that. That's where Octopus Deploy comes into the picture.

Octopus Deploy is a friendly deployment automation tool taking over where your build or CI server ends. Use Octopus to promote releases on prem or to the cloud. Octopus integrates with your existing build pipeline, TFS and VSTS, Bamboo, Team City and Jenkins. It integrates with AWS, Azure and on-prem environments. You can reliably and repeatedly deploy your .NET and Java apps and more. If you can package it, Octopus can deploy it.

It's quick and easy to install and you can just go to octopus.com to trial Octopus free for 45 days. That's octopus.com, O-C-T-O-P-U-S.com.

[INTERVIEW CONTINUED]

[0:15:34.5] JM: Wat about the business side of things. There is this Alexa for business stuff, and I think about a hospital, for example, like transcribing an electronic medical record is really annoying, and having a voice interface in the exam room might be super useful for that. What are the business applications of a voice interface? Have you seen any of these be compelling yet?

[0:15:59.4] PC: I think there are a few. I'll start with the experimental stuff that I've seen some people exploring with and we'll get back into some more things that are happening now. Boston's Children's Hospital, they did a huge thing where they basically took a patient through every phase of their patient life, all the way from they have a concern, to their being admitted, to having a doctor visit, through surgery, through recovery, through at home, all those different phases and they voice-enabled, every piece and they were trying to figure out what would be best with the customer, what would be best for the nurse, what would be best with the doctor.

Boy, they had a lot of interesting things all the way through. Just reducing patient confusion, reassurances, dealing with situations where your hands are completely busy, all those kinds of things. There's a lot of neat stuff there that they did. I think, to bring it back more to what's happening right now, I think that's a little more experimental. What's happening right now is you can use Alexa for business to deploy and configure your devices, so that all the people in your office have access.

The scenario that's killer for me right now in our offices is you walk into a conference room and you say, “Alexa, start the conference.” That's it. Everybody just starts joining in, and you're going, you're on your way. It's really great. The worst experience in that is, I mean, the hardest it ever gets, let's say it that way. The hardest it ever gets as you say, “Alexa start the conference.” She says, “What’s your number?” You the six-digit number, you just say it out loud. You don't have to dial it. It's really cool. Most the time, you don't even have to do that.

The other situation that is looking really interesting there is hotels. You can take these devices and then deploy into all the rooms in the hotel, and because they're centrally managed like you would want in an enterprise, you can say what's on them and what's not on them and what people can use and so forth. You can walk in and open the blinds, turn the lights, all those things, but you can also order take-in, or dine-in. I think there's a few different scenarios that are

starting to play in now. As we said before, it's early. It would be really cool to see where they go from there.

[0:18:01.4] JM: You mentioned this starting a conference with Alexa. If you walk into an Amazon conference room and you're starting a conference and maybe there's some people who are present and then there's other people that are remote and they're calling in, or having a voice video call. If I wanted to build that application, what would be the steps that I would take?

[0:18:23.2] PC: First of all, you would provision your devices. You'd say, here's the devices and you would add users to them and say who's allowed to use this and so forth in what situations. In this case, you set up all the conference rooms. Then you connect it up to whatever you're conferencing system is. We use Chime, which is a video teleconferencing system. There are others that you could use.

Then whenever a person schedules a meeting, they just include the Chime room, or whatever you're teleconferencing system is in the meeting invite, and then Alexa will then know about it. You can go through the Alexa for business site and configure all of that.

[0:19:00.6] JM: Okay, let's go through another example. I want to drive into example and give a breakdown of the design and implementation process, A very simple application, so let's say I want to build a voice application for interacting with Software Engineering Daily content. We've got a bunch of different episodes, we've got different topics of those episodes, we've got different guests on those episodes, so there's some schema to this data. We want to be able to play episodes, we want to be able to find favorite episodes, things like that. How would you recommend going through the design process, the upfront design process where I'm just trying to map out what this application looks like?

[0:19:42.1] PC: Well first, let's talk about two options you have. One is you could just create a flash briefing which would let you play the latest podcast and stuff like that. That's more of just a pure consumption experience where the person can just say, “Tell me the latest,” and you just start playing it.

The one you described is the more interactive, where you can find certain people or topics and so forth. Let's go down that path. The first thing that I would do, I think of design in two phases. Phase one is just write a script, literally just like you would a movie script. What does the person say, what does Alexa say? Was does the person say, was does Alexa say? Now, this of course is the very happy path version. There's not all the possibilities and all the combinations, but what you're looking for is, is this a really interesting good experience and are you getting the value you that you want out? It's really fast to iterate. You can just crank out a lot of variations in this.

I even like to copy and paste what Alexa says into the dev tools and you can start listening to her, and just say, when I wrote it, reading something is different than listening to something, because you can skim and even the way you structure sentences can be a little bit different. Just listening to it at that point makes a lot of sense. That's the first piece.

Now, at that point you haven't figured it all about how you're going to architect your code or anything like that. You just have gotten this experience that you like. The next part is as you start to flesh out all the different scenarios that you want to do, you can start to think about what is it that your skill is able to do? You listed a few things there, like your skill is able to find an episode by guest, or find an episode by topic, or maybe you go a little more complicated and you would go find a answer by a question. I could ask your entire body of work a question and maybe you've already asked somebody about that question and then they could just show the answer from that sub portion of the clip, right?

You'd think about that. What can my experience do? Once you understand that, you can start to line up intents. Let me let me give you a few different bits of terminology here, just so you have all the pieces. There's an intent and this is what your skill can do and this is what customers want to accomplish. For example, search for a topic. Then on top of that, there are utterances and utterances are the variety of ways that somebody might say an intent.

They could say search for a topic on AI, or find me a topic, get me topics on, they can say a wide range of find me a topic. Then the last part is a slot. A slot is a variable, or parameter, or something like that that you would send in. You could have that core part of the utterance be the same, like get me a topic on, or find me a topic on, or whatever. Then the last part would be on AI, or on databases, or on voice. That last one would be a slot.

What happens is the customer can say all sorts of things, that all becomes – all those utterances become training data for the natural language engine. Then what the Alexa service will do is it will send you an intent and an utterance. You're not doing any of the AI parts, other than providing trading data, and then routing all of those phrases to an intent.

Then we get to the part where you could write the code. Now you have an intent, now you can just go through your data and use a web service to say, “Given this topic, here's the response I want to give.” It all becomes this call and response at that point.

[0:22:59.5] JM: An intent is a method. It's like if I wanted to map out the functionality of this application, maybe I would start with an intent, like the intent is search for and list topics. The ability to search for and list episodes within a topic, and then utterances are all the things that somebody can say that will map to that intent. You might say, “Give me an episode on AI, or tell me the latest AI episodes,” or all these variety of ways that you could express the same thing. Then slot is the variable. It's AI in this example, or maybe you could even have AI map to the same things that machine-learning map to within the intent implementation.

[0:23:46.3] PC: That's right. Yup. You've got it.

[0:23:48.0] JM: Okay. Let's go top-down again. If a user says something to the Software Engineering Daily voice app that hey, maybe somebody out there in the audience, if you want to build it, please build it. We've got an open source project. I would love to see this thing be built, that would be cool. Anyway, the phrase, so if I say something to the app, if I say, “List me the latest episodes on AI,” that phrase gets interpreted into text, and then the application has to be able to interpret that text. Help me understand a little bit more about how –

[0:24:21.4] PC: Yeah, so here's how that works. What you'll have is a web service and your web services waiting for a request from the Alexa service. What'll happen is you'll get this request, this is JSON file, you get this JSON request. Inside that request, it includes, “Hey, this person just asked for a search intent and they wanted a list episodes intent, and they particularly wanted to know about AI.” It would have an intent of lists episodes and the slot of of AI.

You now have that request and then you can run your logic, whatever you would want to do in your own web service, so that's probably you do a database lookup, or an API call and you using the slot information, you reduce down to that list, and then you compose what you want to say. You would compose a sentence, or two, or maybe even some audio files like [inaudible 0:25:13.0] found them, or whatever you want to do. You would send that back to the Alexa service as a response. You get a request from the Alexa service and then you compose your own response and send that back. Then Alexa will say it. She renders the text back to the customer through voice.

[0:25:32.4] JM: What about context? Now that I've asked the Software Engineering Daily app to give me a list of the most – the AI episodes, is that context sitting there in the device? Can I now say, “Okay, I want to select episode number five from that list that you have presented to me?”

[0:25:51.6] PC: The way it works is it's not really sitting in the device, but is a part of that request and response. Well there are a few ways you keep memory. Let's talk about the most basic way, which is the scenario you just laid out, which is within my session, so once I start a session with the skill, we can go back and forth and keep having this conversation, and I can keep track of the content that's going on. I could say the context that’s going on. I could say you're looking at AI sections. Here's the list of five. Let's play the first one, you say, “Play the next one,” and I know what the next one is, because I have that context. That's transferred back and forth and the requests and responses.

The other way you can do it, if you want to have a longer term memory as you can store that off into the database. You can say, let's say they go away and they come back the next day and they start the skill, you could just say, “Do you want to pick up where you left off? You're in the middle of episode three,” and then you can pick them up and go. That's a longer term memory is where you store those session attributes off into a database to recall them later.

Yeah, absolutely and that's an important capability that sense of memory, because that's what converts a skill from a simple call-and-response, like set the timer and so forth into an actual conversational UI.

[0:27:00.9] JM: Is the user able to configure what data gets stored on the device, versus getting stored in the cloud? Because you can imagine certain applications where low latency response is super important and other applications where yeah, it's fine if this stuff is mostly in the cloud.

[0:27:18.9] PC: The way it works is very little is actually happening on the device. That's how we can take Alexa and put her on to all sorts of devices, like Raspberry Pis, or into – you brought up home appliances and things like that put in cars. Very little is happening on the device. Really what's happening there is the wake word detection and the signal noise reduction with the far-field microphone arrays and so forth. All of the logic is happening on the web services.

[0:27:46.0] JM: Yeah, it will be interesting to see how this develops, because I had a conversation with somebody on the Amazon machine learning team, where they were talking about machine learning at the edge and how this difference between edge devices and the cloud, it's going to change over time in a – eventually will – I think right now, since its early days, it's totally fine that everything occurs in the cloud. Over time, we will need more low-latency models and it'll be interesting to see how the trade-off between processing and storage in the cloud, versus on device will evolve over time.

[0:28:24.3] PC: Yup, it will be. It's interesting times for sure.

[0:28:28.2] JM: The design of a voice app, this isn't a new paradigm. What are some common design mistakes that people make?

[0:28:35.4] PC: There's a few. One of them is taking what are clearly best practices on web and mobile and just bringing them over without much thought to how their application is used. Here's an example, on a webpage or a mobile site, you really think about consistency, because as we've discussed a little earlier, it's all about how easy is it to learn and consistency is just a great tool for that. If everything is the same, it means that I can skim over the page and ignore the parts that are the same, or quickly recognize the parts that I've already learned and just reuse them.

With voice, that same strength can be a liability, because if you hear the same phrase over and over again, you just start thinking, “How do I skip this?” There's no sense of skimming in voice. You get it all in a linear fashion. Adding variety is actually an important way to bring that forth. A simple version of that is whenever you start the skill, have a different welcome. Welcome, hi, how you? What's going on? Those kinds of things. That's a simplistic way of thinking of it. As you work into more complex versions, you'll start to bring in different ways to phrase your answers.

If your skill is very much around showcasing some numbers, sometimes you might say the number is this, other times you might say, “You know, we've seen a trend of this number trending up and it's now reached this moment, right?” Those kinds of things you start to bring in a little bit of variety. Now you still want to use consistency in some situations, particularly where you're trying to teach people exactly what to say. You might use a key phrase, or something like that in your response. The spirit of learnability is absolutely still at the core. Your tool of consistency should be used a little more sparingly, I suppose. That's one example of how people think of it differently.

[0:30:23.9] JM: Tell me something unintuitive about the way that people interact with voice applications.

[0:30:30.6] PC: Something unintuitive with it. There's a couple different ways that you might think about this. There's a few surprising scenarios, I guess that I've seen. One was I've seen a lot with both youth and elderly who can approach the device and they just use it in a unencumbered way. In fact, they may not have really experienced the web, or mobile very much and they just have a more natural approach. The delight that they see, you see in them is pretty cool. That sort of not trying to talk to your computer like it's a computer. A lot of people try to reverse engineer how things work, so they can more effectively make them work. Having folks just be able to say what's on their mind and get responses back is pretty cool.

Similar to that, this design pattern of you've probably seen those voice experiences with banks, or call centers to support centers and things like that, I think the more modern ones of those are actually getting really good, but there's been years and years and years of training for

everybody just to listen for the number, or to say operator, operator get me a representative, those kinds of things.

It's hard for people to break out of the mindset of what the new ones are probably able to do and what Alexa, the whole approach of Alexa, which is not that IVR-ish style approach. It's a different backing technology, I guess.

[0:31:53.8] JM: Are there any aspects of mobile application design, which is what we've been very used to for the most part? Is there anything that we need to consciously throw away when we're coming to the world of voice application design?

[0:32:07.9] PC: The thing that I would love to throw away is the concept of flow charts; very rigorous, rigid, structured arrangements. Because what happens is people often do over answering and they answer something a little bit differently than you would expect, so here let me give an example of what would be great to be built with a wizard, or a very flowcharty way in mobile, but just really doesn't work the same way in voice.

In mobile, you might say, “Hey, I want to plan a trip.” Then Alexa would say, or the app would say, “What's with Alexa's experience?” Alexa would say, “Okay, where do you want to go?” The person could say, “I want to go to Portland.” “Okay, when do you want to go?” “Next month.” “What do you want to do?” “Kayaking.” Right, all that would be fine and that would work in both mobile and voice the same way.

If I do it again and I say, “Okay, where do you want to go?” The person says, “Kayaking,” well that's a little bit different, because they gave you an activity and not a destination, right? “Where do you want to go?” “I want to go kayaking.” We shouldn't then next say, “Okay, well what would you like to do when you get there?” That would be a pretty bad experience. We should actually know that that's an activity, versus a destination.

[0:33:08.7] JM: How about kayaking is a beautiful place in California.

[0:33:11.5] PC: It totally could be. Kayaking in California, that could be. Another thing you might say is I want to go to Portland to go kayaking. In that case, you over answered. You wouldn't

want to then prompt to the next question. In a website, in a wizard, you wouldn't really type both answers into the same box, right? That's where the design of your utterances and slots come into play, so you can understand which field to go where, and you're not stuck in this rigid I must get the answer that I want next. You instead, what you do is you use a – instead of a graph-based design, it's called a frame-based design.

The idea there is you say, “I have this collection of stuff that I would like to get and whatever the customer says, I'm just going to take as much as I can and fill in as much as I have. Then I'll prompt for the stuff I don't have.” You just keep in that loop, instead of going through this very rigid path.

Another way to think of that is, maybe a simpler way to think of that is with a menuing system. Menus are also another rigid path, like find your way to something. With a banking experience, you might say, “I want my routing number.” In a mobile app, the absolute best design would be to say not put the routing number at the top level, because most people don't need it that often, but when you do want it, you want to know how to get there. You go menu, menu, menu, menu, menu, routing number. It's the information architecture that makes it very easy for you to find that.

In voice, it would be pretty bad if you said menu, menu, menu, menu, what's my routing number? They wouldn't be intuitive, or anything. In a weird way, what you do is take your information architecture, they can be very tall and deep in a mobile app, or a website and you flip it on its side, so it becomes very wide in a voice design. Now you have routing number at the very top level. You say, “What's my routing number?”

It's a bad idea to do that in mobile, because routing number would just take up valuable pixels, it would add conceptual overload to that main screen. In voice, it doesn't do either of those things. Rather, it actually helps with discovery, because if you just serendipitously try things, like what's my routing number? Well, it's there for you. A way of thinking of that information architecture and more of a less rigid way and more of a way where people can just say what's on their mind and you'll catch it.

[0:35:20.9] JM: I want to go back to that example you gave with the kayaking. You say, “Alexa, I want to go on a trip.” It says, “Great. Where would you like to go?” You say, “I want to go kayaking.” Then Alexa probably is not sure what to do, unless kayaking is a location in California, in which it's going to probably give you a response that you don't really want to. It's going to say, “Okay, planning your flight to kayak in California.” That's not a great response.

It seems there is – this is a situation, where you would probably want to, I don't know, throw an exception, or have Alexa make some log message, so that the developer knows that erroneous behavior, or deviant behavior has occurred, so that the developer can accommodate for that situation. How do you accommodate for those educate – because there's obviously all these things that you're not going to expect when you're dealing with this new type of interface?

[0:36:08.9] PC: Sure. Let's handle that one first, because I think that's one that you would handle, and then let's do one that you, like that's clearly more, or you’ll say, “I'm going to go to crayon,” or something like that. In the kayaking example, the skill is about helping you find a place to go and things to do when you're there. It absolutely should have an activity slot full of activities. I would include, maybe in your situation kayaking, if that's one of the activities that they do.

In that case, what would happen is when you get the response back from the Alexa service to your to your web service, it would include activity equals kayaking. You would go, “Okay, cool.” Now I know the activity is kayaking. Instead of saying, “Where do you want to go?” You might say, “What city would you like to go to?” You get more specific on the requests. That's very easy to do, because what you do is you have a location slot, which is to fill in your, or a destination city slot, probably in this case. You would have, your first prompt would be where do you want to go? Then your next prompt would be, what city do you want to go to? You can get more and more specific and you can phrase them in different ways and it just works down the list, until it gathers the information you need.

Kayaking is a good example. Now maybe you say, let's go one level more of error-correcting, which is let's say, you handle watersports, but you don't handle kayaking specifically, right? You might say, “I'm going to do –” you would have a top-level slot value of watersports, and then for that you would create synonyms, like jet skiing and paddleboarding and kayaking and so forth.

Then what would happen is the Alexa service would say, “Hey, the person said kayaking, but they meant watersports.” You get both back. Then as a part of your response, you can say, “Okay, there's a few things like – we have activities that are a lot like kayaking. Are interested in watersports?” You could do an explicit conformation like that.

Or you could just say, “I found some watersports for you. When do you want to leave?” You can move on to the next question, and you can do an implicit confirmation. That whole part would work.

Now let's do the case where they just do something really outrageous. You say, “Where do you want to go?” They say, “Dog,” and you don't know what to do with that. In that case, what you could do is you could log off the value of dog and put it into your instrumentation data so that you can say, “Oh, a lot of people are asking for this city, or activity, or whatever it might be,” and you could build a scenario that helps them. That's a way that you could add more capabilities to what you're doing.

The other thing you can do is you can use something we have called a fallback intent, which basically is anything that's outside of your voice model will get caught by this. Then you can say, “Hey, we don't quite know what you're talking about.” You can build clarity. You can start to say, “I'd to know more, ask a more specific question.” That’s another way you can sort of air handle.

Segment allows us to gather customer data from anywhere and send that data to any analytics tool. It's the ultimate in analytics middleware. Segment is the customer data infrastructure that has saved us from writing a duplicate code across all of the different platforms that we want to analyze.

Software Engineering Daily listeners can try Segment free for 90 days by entering SE Daily into the how did you hear about us box at sign-up. If you don't have much customer data to analyze, Segment also has a free developer edition. But if you're looking to fully track and utilize all the customer data across your properties to make important customer-first decisions, definitely take advantage of this 90-day free trial exclusively for Software Engineering Daily listeners.

If you're using cloud apps such as MailChimp, Marketo, Intercom, Nexus, Zendesk, you can integrate with all of these different tools and centralize your customer data in one place with Segment. To get that free 90-day trial, sign up for segment at segment.com and enter SE Daily in the how did you hear about us box during signup.

Thanks again to Segment for sponsoring Software Engineering Daily and for producing a product that we needed.

[INTERVIEW CONTINUED]

[0:41:29.1] JM: You mentioned that you don't like the flow chart model of developing software for a voice interface. One thing I have heard you recommend is the idea of making a script. You want to – just like you would write a movie script for how people are scripted to talk to one another, you would write a script for how the user and the Alexa skill are going to interact. This to me seems in some ways like a flow chart, because there's going to be branching points, right, where Alexa asks – the Alexa is, “Where would you like to go?” There is going to be a point where the user might – or, “What would you like to do?” The user might say, “Go to Oklahoma,” or, “Go kayaking.” It seems those might branch off into different intents. You would want the Alexa to translate those into different intents. Help me understand, what goes on in that script design, since we're really focusing on the top-level app designer. What is that script design experience like?

[0:42:32.8] PC: Yeah, and let me start with what I think most people are doing right now is flowcharts. It's very comfortable, it's easy to get into that, it just has some pitfalls, where you start to get too rigid. If you start coding to the flow chart, you end up with an experience that ends up being very stilted. Here's what I think is the remedy to that, but it is a different way of thinking, so it's a little bit harder to get into. The first is the script part. It's a very linear thing. You just go through a path and you just go after your golden paths.

In the script portion, I just avoid the branching. If I think there's another line of thinking that is significant, I'll just make another storyboard for it, so you start moving into storyboards. I might say, “This is my first time experience storyboard,” or this is the case where they keep getting the answers wrong storyboard. What you're trying to do is just exercise all of the logic that's going on behind the scenes.

The way you handle the branching scenarios is more about handling the situation. It's a situational design. It's where you get a couple things before your logic has to run. One is you get the intent and the slots and so forth from the customer, and the other thing you have is the current state, so you know what's going on in the system. Like in a trivia game, you get an answer from the customer and you get the current question. Because you have those two things, you can form a response and you send it back. It's very much a call-and-response style system based on those two pieces of input.

Unlike a mobile app where it's more of an object-oriented UI, where you interact with objects, or more of a game UI, where there's a game loop running. It's more of a call-and-response experience. That situational thing is where you get – where you're able to get rid of the branching. A very common place to start introducing a flowchart would be in your welcome experience, where the first time through you might want to be a little more verbose and explain what your skill does and so forth. The second and third time, you're getting a little, more right into the meat of it. Then the fourth, fifth, sixth time, maybe you're saying, “Wow, congrats for coming back six days in a row,” and you're doing some, like streak celebration, that kind of stuff.

You could imagine doing a flowchart to lay all those out, or instead you could just say, “Well, I know the intent was a launch request and I know the situation is they've been here four times.

Here's my response.” It's more state-machine ask, where you've got these situations. The getting really rigid on the flowchart gets to be a problem when they don't answer the way you expect them to, right? It just brings up more error cases that you have to handle less elegantly, I suppose.

[0:45:13.0] JM: Yeah, I can imagine. When we're talking about building voice applications, tooling is important. My IDE has helped me write clean, well-formed Java code. What are the sets of tools, or frameworks that are available for building voice applications?

[0:45:31.7] PC: Yeah. There's a few and they're all at different levels, I suppose. There's tools inside of our dev portal that help you capture, collect and design that interaction model, and the interaction model includes all the intents, utterances, slots, the synonyms, the different prompts that you might want to use, all the variety, those kinds of things. That that will help you configure that entire how the customer, like what the customer says. They say these things and then those all resolve down into these intents that get packaged up and sent to you. That's the interaction model design. There's a bunch there.

For the coding side, we have SDKs that will help you with a variety of languages. We have Java and JavaScript, or node really. Then we also have some samples in C-sharp and Python, but again, you can use whatever web service language you want. We have these SDKs that will help you. They in particularly help you parse the requests and compose the responses, and they help you do it in a way that is aligned with this situational design around getting both what the customer says and knowing the current situation, the current state and then responding. They have this notion of, “Hey, I can handle this situation. Now here's the response.”

A little bit deeper than that, the Alexa service itself has a few different things that help you. I mean, the first is completely under the covers, all the natural language understanding, ASR all that stuff, you don't have to do any of that. There's a few other pieces, the dialog management stuff, where you can compose that whole, like anytime you want to collect information in a form or through a wizard, you can use dialog management. In dialog management you just say, “Here's what I want to get,” and Alexa can handle the entire thing for you. Or you can watch every step of it and you can override things and do as you want throughout the conversation. The dialog management is a pretty powerful one.

Then another one is called entity resolution. That's where you can transform synonyms into the values that you want, so there's tooling for that too. Then one more, last one is the command-line interface. That's where you can go into terminal and you can just start building everything from there. You can test your skills, you can deploy them, you can use your own IDE as much as you want, and then as you're coding along, you just – deploys your lambda function, and also your interaction model.

[0:47:45.8] JM: Have building these voice applications made you rethink how you talk?

[0:47:51.0] PC: They have. Yeah. Even as I'm writing now, I see different things. One of the examples was this – we use, in Amazon we read a lot of these six-pagers, which describe how are things going to work and what we're going to go do. As a part of that, we used an Oxford comma, which is if you have a list of items, you'll say thing one and – or thing one, comma thing two, comma thing three, comma and thing four. You put a comma and an and. That way, you know that the last two things aren't grouped together. They're separate items. I mean, it's like an Oxford comma separated list.

I was listening to NPR and they were talking about these owls that were all over the world. They said these owls are in North America and South America, as well as Europe and Australia. They use the technical clustering there, where they clustered the first two items with an and between them, and they said, "As well as," and then two more items with an and between them. Connect you might with a phone number. You say 247, 47, or whatever, right? You sort of cluster numbers together. I guess, I've just noticed a lot more places where I noticed how I skim and how I phrase things that are a little bit different in a verbal way than they are in a written way.

[0:48:57.8] JM: Amazing. What about designing the audio and visual experiences? As we start to get towards a world where you have visual devices that also have audio capability, how is that space being explored and what are the common design themes you're seeing?

[0:49:15.4] PC: Yeah. I think there's a spectrum there. There's one model is that you have a very voice-first experience. That's where we're talking and we might have a whiteboard between us, and I'll be taking notes on, or maybe a napkin, we're inventing something together. We'll be

sketching on that. It's facilitating the conversation, but the conversation is still the lead, right? It's what the conversation is what it's really all about, and the visuals are just there to set anchors for our memory, or to clarify a point, so that we can iterate on that. It still is this very cooperative experience, where we're trying to do this task together, either invent this thing, or transfer some knowledge back and forth and so forth.

With a screen device that is like a mobile phone, or something like that, there's always this risk that you're going to fall back into a traditional mobile app, or traditional web style design, where it's not a cooperative experience. Instead, the app is saying, “Here's how you use me. Use me this way.” Get your job done, get your task completed. Then in that world, I guess voice could be relegated to a helper thing, like a keyboard, or something like that.

For me, that part is less exciting than the part where we push out into this more conversational space. We get to some of stuff you were talking about earlier, where you can imagine a future 10 years from now where you have a combination of sensors and screens and so forth that form this ambient computing experience, where it's more about the conversation at the heart. Yeah, there's definitely some gravity there, because that's where the world is today is this screen-first, we lead you through an experience. I would love to see us to get to a point where this conversational experience takes the center more communal, more with the people around you and your computing, rather than just you and your computing.

[0:51:01.9] JM: Definitely. To wind down, so I think of this Alexa project, it's a three-sided marketplace. You have to get Alexa in devices, you have to get developers writing apps for those devices for the Alexa interface, and you need consumers that are using it. Amazon as a company is trying to grow all three of those constituencies, you specifically are focused on growing the developer side of things. What are the techniques that have been the most effective in getting developers interested in this platform?

[0:51:34.0] PC: There have been a few. I mean, we are really just looking to take the great ideas that developers have and help them get to customers. We do that in a few different ways. One is we've done hackathons. It's a way for developers to get really, really creative and do stuff that may not even ship, quite honestly, right? It's really out there. They do it over the course of 24 or 48 hours. Those are fun. We do a lot of workshops, we do two-hour, four-hour, even eight

hour. We have this series called dev days, where you can go and learn about voice design. It's more of a classroom, lab-oriented setting. That's mostly a few minutes of concept and then more minutes of let's go build something together, and then some more concepts and some more building; it's very hands-on kind of a thing. It's pretty fun.

There are a few different kinds of talks that we do. We'll go to big conferences and just try to work with the decision-makers and stuff to let them know what's possible. Lately in the last month really, but really for this year, I suppose, but last month has really been ramping up. We've been doing a lot of live-streaming, so on Twitch TV we have this Amazon-Alexa channel, where we're just building skills and some with the gaming buttons, some with monetization, with insta purchasing, some they're very focused on this design concepts we've been talking about, others on the tooling, like here's how you use the SDK and so forth. Yeah, there's been a lot of different approaches like that, a lot of meetups.

There's a set of champions that are fantastic. There's a 25 Alexa champions who are around the world and they are just on the bleeding edge of what's happening and they know the latest and they work with our communities to both help them understand what's possible and then also to hear what's not possible yet. Then they come back to us and say, "Oh, man. There's some stuff that we'd love to see and do that feedback loop." Just really going after all the different angles we can.

[0:53:19.6] JM: Very cool. Well, Paul Cutsinger thanks for coming on Software Engineering Daily. It's been great talking to you.

[0:53:23.1] PC: Thank you, Jeff. This has been great.

[END OF INTERVIEW]

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We're proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]