**EPISODE 555**

[INTRODUCTION]

**[0:00:00.3] JM:** The Ethereum and community started as a small group of dedicated engineers. It has ballooned to thousands of engineers, entrepreneurs and investors, all of whom have a stake in the direction of Ethereum. Ethereum theorem is an open source project and the direction of a popular open-sourced project can get complex. Ethereum is figuring out how to govern itself, and it's not clear what the perfect model is, but there are a few historical examples to think about, namely; Linux and Bitcoin.

Linux is similar to Ethereum and that there is a clear leader. Linux has Linus Torvalds and Ethereum has Vitalik Buterin. Linux is massively successful, and the Linux development team does have a top-down hierarchical approach. But does a hierarchy with clear leadership makes sense for a project like Ethereum, which has decentralization at its core?

Bitcoin is headless. Satoshi disappeared in 2010 and there is not an official leader. Bitcoin has succeeded without a well-defined hierarchy, depending on what your definition of success is. Bitcoin development does not move as fast as a Ethereum, and this is by design, but there is more widespread trust that the integrity of the Bitcoin system cannot be compromised by its sole creator.

Hudson Jameson is an Ethereum developer and entrepreneur who has been part of the community since the early days. He works on Ethereum governance, which defines how changes to the Ethereum project are proposed, accepted and implemented. Hudson joins the show today to talk about Ethereum governance and smart contracts and the DAO hack. We did not discuss on-chain versus off-chain governance, but I am hoping to cover that in a future episode.

Before we begin today, I have an announcement. We have launched Software Daily, which is a place to post your software projects and discuss them with other people. On Software Daily, you can find collaborators and feedback for your software project. If you have an open source application or a side project that you've been tinkering with, or an academic computer science

 

paper that you want to get feedback on, then come to softwaredaily.com and post what you're working on.

Software Daily is about cool projects, new ideas and creativity. If your project is especially interesting, we'll send you a Software Engineering Daily hoodie or a t-shirt or even have you on the podcast to discuss what you're building.

I've been posting some of my own side projects on Software Daily and I'd love to see what you're working on. Come to softwaredaily.com and discuss your ideas with the community. We'd would love to have you there.

[SPONSOR MESSAGE]

**[0:02:54.4] 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, although I've gotten better at talking about software engineering, I have definitely gotten worse at actually writing code and doing software engineering myself.

But if you want to check out that quiz yourself and help us gather data, you can take that quiz at triplebyte.com/sedaily and in a few weeks we're going to take a look at the results and we're to find out if SE Daily listeners are above average. And if you're looking for a job, Triplebyte is a great place to start your search, fast tracking you at hundreds of top tech companies. Triplebyte takes engineers seriously and does not waste their time. I recommend checking it out at tripledbyte.com/sedaily. That's triplebyte.com/sedaily.

Thank you, Triplebyte, for being a sponsor.

[INTERVIEW]

**[0:04:25.2] JM:** Hudson Jameson is an Ethereum developer and entrepreneur. Hudson, welcome to Software Engineering Daily.

**[0:04:29.8] HJ:** Thank you for having me.

**[0:04:31.6] JM:** I want to talk about a variety of different things today; Ethereum governance, your entrepreneurial efforts. Let's start a little earlier though. when did you get into the Ethereum community?

**[0:04:42.6] HJ:** So I guess predating Ethereum, I got into Bitcoin and Blockchain around 2011. Spent a few years in different communities, including the Dash community back when it was called Darkcoin. Then after that, out of college, worked at a banking and insurance company leading their blockchain program that was called USAA, or the company was called USAA rather. After working with them, I decided to leave in the summer of 2016 to work for the Ethereum Foundation.

So I guess the first time I heard about Ethereum was around Vitalik put out the whitepaper and when they were announcing the presale. I just thought it made complete sense to have these turing complete programs on a blockchain. So it just kind of clicked with me and I was obsessed ever since then.

**[0:05:31.1] JM:** In the pre-Ethereum days, when you were tinkering with Dash, or Darkcoin as you said it was called then, what was it like back then when you were tinkering? Did you feel like the communities were a little more disjointed and fractured than they have become with Ethereum?

**[0:05:46.2] HJ:** Yeah, I guess. It kind of depends. Back in the really early days of Bitcoin, it was definitely less political I would say. There was definitely some disagreements, but everyone was somewhat working together at least. Then as far as Darkcoin went, it was a similar thing up until they decided to rename it to Dash. When they decide to rename it to Dash and then the years after that it kind of — It's been very successful from what I understand. I don't follow it as much anymore, but they seem to have some like disagreements around the time they were doing the renaming, and I haven't kept up with it much since then.

**[0:06:23.4] JM:** When the Ethereum community was just getting off the ground, you believed in the turing complete vision of Ethereum itself. What do you think of the community? What was your perspective on the talent and the implementation skill and just the overall [inaudible 0:06:40.5] of the community?

**[0:06:42.3] HJ:** So back in the really early days, it was a lot of people like me who were really excited about the idea, but the thing that kind of sets the community for Ethereum apart and still does today is the fact that Ethereum as a blockchain technology doesn't rely on the single use case of value transfer and value storage just like Bitcoin and a lot of the other altcoins that were around at the time. This was one of the first forays besides maybe Mastercoin and a few others into either a layer on top of the blockchain or complete blockchain that had programs on it for use cases beyond just money, or money transfer, or money storage.

So that brought a different class of people in that were much more tech-focused I would say, and that's really what makes a difference even today, is the fact that you can have so many different applications and uses on Ethereum and software like Ethereum that it attracts a different class of people, different type of people than just people looking to — Attract less people looking to get rich. I mean, obviously that's very different now with the ICO era we're in, but at least back in the early days I feel like it was a lot different just because we have more proliferation of potential use cases.

**[0:08:01.4] JM:** I remember, I was talking to some Bitcoin maximalists around the time that Ethereum launched and they were very skeptical that this would ever work. Do you remember addressing the skepticism back then? I mean, you are so early on, you yourself probably had a little bit of skepticism or were you just wide-eyed and optimistic?

**[0:08:22.8] HJ:** I definitely had skepticism just because — I mean, there were a lot of things that had to go right in my mind for it to be successful. Things that I didn't completely understand, a lot of economics around it. Things I still don't understand today. I mean, now the term is crypto economics, but getting that right and getting things like gas metering. So how do we make it so that you can't attack the system by having an infinite loop of a program running? So getting the gas values that you use on Ethereum, gas being the kind of fuel to make sure that you don't run

a program forever. Getting that right early on seemed important as also the measures to prevent people from wanting to attack the network, making the — I think it was called Dagger-Hashimoto back then, but I think it's may be called something now the type of consensus protocol and the changes and using uncles and all that other kind of neat stuff that was the first of its kind back then. It was just all things I was keeping an eye on, but I was definitely still skeptical as a lot of the other people in the community were that this wouldn't go anywhere. Then I knew it was spawn other things. Even if it failed, it would spawn other things that would be successful around the same ideas of having a turing complete blockchain language.

**[0:09:44.0] JM:** You mentioned gas a couple of times there. We've explained gas on the show in several different episodes, but I think this is — To my mind, this is the hardest aspect of Ethereum to get, or perhaps one of the hardest. Explain what gas is just so we can give people another chance to understand it.

**[0:10:03.9] HJ:** So, in Bitcoin, you have transaction fees when you want to send coins from one person to another. Same thing with any traditional, like wire transfer or other financial system. You're paying that third-party to perform an action for you. Well, when you have a blockchain system, the third party or the middleman, middlewoman is the blockchain itself. It's the computation itself. So where as you would pay a transaction fee for transferring value, you instead pay it to the network of miners who are mining the coins for you. So that's kind of Bitcoin in a nutshell.

What a Ethereum did was it took it a step further and said, "Instead of just transferring a crypto coin, like Bitcoin or ether, we're to run programs on there." So what they had to do was find a way to make a system where you could write programs on there without having them run forever. In software development, there's a thing called infinite loops where you have a program and it always just goes back and reactively ask itself the same question over and over and over again, and it can take up a lot of computing power, which there's programs that detect those and then when they detect them they put a stop to them.

So Ethereum implemented that using gas economics. Whenever you do a program on the Ethereum blockchain, you have to kind of pre-calculate how much it's going to cost to run that program. Every single computational step in the program such as an addition, subtraction,

hashing, if you're going to store a value, each of those are priced out. So maybe some of the more simple operations like addition might cost like, let's say, one gas, but something like doing a SHA256 hash or maybe storing the value or doing a complicated check could cost in the thousands or hundreds of thousands of gas.

Gas is kind of the way that we value how much a computation is going to take or how much computation that an action is going to take, and then you pay for gas using ether. So you can think of it as, “I want to run a program that calculates one plus one equals two.” I'm like, “That's going to cost three gas,” and as someone launching the program I say, “Okay. Miners, I'm willing to give X-amount of ether per gas.” So let's say .001 ether per gas amount. So there's this competition on the network where miners will select the most profit or the miners will choose to collect the most profit by choosing the smart contracts that have the highest cost gas calculation that someone put out as an offer.

If I have a program that's one plus one equals two and I say I'm going to pay .001 ether, but someone else says I'm going to pay .002 ether to launch it, then the .002 ether person is more likely to get that transaction mined. As the program runs, it consumes gas, and once the gas is consumed, if all the gas is consumed before it completes, it reverts the whole program like it never even happened. If you don't use all the gas, then it refunds you the remainder of the gas.

[SPONSOR MESSAGE]

**[0:13:38.2] JM:** GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use, and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plug-ins. Use the value stream map to visualize your end-to-end workflow, and if you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

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 longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[INTERVIEW CONTINUED]

**[0:14:59.6] JM:** So just if I can give a crack at re-explaining what you just said. So you've got these smart contracts that are written in perhaps a language like Solidity. Solidity compiles down to EVM byte code and each of those EVM byte code instructions has a gas price associated with it. Is gas price the term to use there for an —

**[0:15:21.3] HJ:** Yeah, gas price associated with each opcode.

**[0:15:24.5] JM:** Right. So there's a gas price associated with each opcode, which means that your Solidity smart contract is going to compile down to a collection of these EVM byte codes and the total price of the contract is going to be the sum of those EVM instructions. Then when somebody wants to call that smart contract, because they're going to have to call the smart contract on every full node that is on the Ethereum blockchain, because every full node is running that smart contract and every full node is going to have to execute that smart contract to maintain the same state, then people who want to call that smart contract, they issue their transaction to call the smart contract with a gas limit, I believe is term, and the gas limit is essentially the proportion of ether per gas that they're willing to pay for the smart contract execution. Is that right?

**[0:16:19.1] HJ:** Yeah, that's right. A couple of corrections I would say is that the ether that you're paying to actually run that to either launch or to interact with the smart contract is going to the miners only. If you are a node on the network, you're not getting that reward. You're just emulating those transactions through the complete history of the blockchain. The miners are actually mining and processing the initial computation that happens, but then it's added to the blockchain and then everyone actually replicates that.

Then the term for the amount of gas that you have to use is just, I guess, amount of gas. The gas limit is another feature of the Ethereum blockchain that's in blocks itself so that you can have a variable block size. Because these contracts — We don't want an attack vector to be that someone make a giant contract that — Or a bunch of small contracts that all can fill up the blocks and make them unusable. You can have a flexible block gas limits so that the amount of gas you can use per block might be 3 million or 4 million units of gas across all programs being executed or created at the time that they're launched. So you can only have so many programs fit in a block at a time.

**[0:17:40.2] JM:** Okay. A gas limit, that's like a feature of the global blockchain itself.

**[0:17:45.2] HJ:** Correct.

**[0:17:46.0] JM:** Okay. So when I'm executing my transaction to call a certain Ethereum smart contract and I know that there's that variable that you put in that is like the amount of — Is it amount of Ethereum and that you're willing to pay? Sorry. That's the amount of —

**[0:18:00.8] HJ:** Yeah, amount of — It's a formula that's amount of gas that the contract requires times the amount of ether you're willing to pay per unit of gas.

**[0:18:10.0] JM:** Right. Okay. The reason that you have this proportional situation where every smart contract has an amount of gas that it takes to run, and every execution of that smart contract you're going to bid an amount of Ethereum that you're going to pay per gas unit. The reason for that is kind of because the ether price changes so much. So you couldn't just have an amount of ether that you would be willing to pay for a smart contract execution. You want to have this proportional situation that allows for some dynamic pricing overtime. Is that the reasoning behind gas?

**[0:18:48.0] HJ:** Yes, that is absolutely part of the reason behind gas. I'd say that's actually the biggest reason, because if you just had it in price of ether, then there wouldn't be that dynamics. Like you said, the dynamic of cost per operation and the fluctuating value of ether itself.

**[0:19:05.6] JM:** Okay. Wonderful. So how would you describe your current role in the Ethereum ecosystem?

**[0:19:12.3] HJ:** So right now I work for the Ethereum Foundation. I have a number of roles there including some community management, moderating the subreddit chat rooms, just kind of connecting people and projects to each other. Some of my other main task are some dev ops work and I edit EIPs. So EIPs are a Ethereum Improvement Proposals. It's similar to the Bitcoin Improvement Proposals or BIPs. So if you hear me talking about EIPs, that's kind of what I shortened it to, EIPs. What I do there for editing is there's a group of us who go in and make sure that the EIPs are being properly submitted. The EIPs are just changes to the Ethereum ecosystem either at a he protocol level or at a community guideline level, like the ERC20 tokens, for instance.

So just being an editor for the EIPs, and then every two weeks we have a core development meeting across client teams. Something that makes Ethereum stand out amongst other cryptocurrencies is that we are not based on implementation, but rather a specification. So whereas in cryptocurrencies like Bitcoin, there is only Bitcoin's core C++ client and then everyone kind of copies off of that, develop the specification.

We have a specification or we're actually developing now a series of different specifications that all align, then we can create clients off it. So we have a C++ client, a Google Go client, a Rest client, a Java client, etc., and each one of those are in different organizations potentially with different teams. So every two weeks, in order to update each other on where the protocol is heading to plan hard forks, etc., we have a core developer call that I moderate and record and put on YouTube every two weeks.

**[0:21:09.0] JM:** I want to get into governance eventually, but since you've been in the community so long, I just want to ask you some other contextual questions. One, you mentioned the number of different clients. I had a conversation with Christoph Jentzsch who was an early Ethereum contributor. He's a member of Slock.it, was eventually involved in the DAO. One of things he did early on was he made all of these tests for different clients to make sure that the different clients would have consistent outcomes, because it would be really bad if you were running a Go Ethereum client that had different outcomes versus the Rust Ethereum client.

Explain why it is hard to keep these different clients in sync and how you can end up with these problematic situations where different clients have different outcomes?

**[0:22:00.9] HJ:** Yes, that is a very, very difficult thing to kind of handle within an ecosystem such as ours with different clients. I'd say that in the last year the situation has improved greatly. We have a testing lead named Demetri who works for the Ethereum Foundation and he does a lot to coordinate and develop a client agnostic test suite so that whenever there are changes to the protocol, some core changes, they have a test created for them and those tests can be loaded into different and the clients and the clients can see how they respond.

There's also a program called hive, which does a similar thing where it runs a series of automated tests. So there's basically — I'd say there's three categories. The first one would be the test I described that Demetri does that are more manual, and then there's hive that takes those manual tests and does some automation around them, which is a tool that was custom-built. The third kind, which is the most fascinating kind of tests is fuzz testing.

So fuzz testing is just a general computation term meaning we're going to throw a bunch of random numbers and data and just crap at a system and see what breaks. It's just like throwing a monkey wrench into a bunch of years, like just seeing what breaks, seeing what's stronger than other components. So we've actually found a number of vulnerabilities from fuzz testing frameworks that go through and try a bunch of random inputs and outputs on different Ethereum clients. We're getting a more and more robust fuzz testing environment, which I think is really exciting, because I don't know of any other projects that really focus on that aspect of testing.

**[0:23:45.9] JM:** My browser that I am running right now on my laptop computer, I believe it's Chrome. I believe it's written in C++ and JavaScript mostly. Maybe there're some other little pieces of language in there, but as far as I know it's only been written in those languages, like there is not a Go Chrome version. So why is it the case that with Ethereum of you have all these different clients that are written in different languages? Why is that useful?

**[0:24:14.6] HJ:** So the usefulness around that is that it creates less of a chance of attacks on the network. For instance, if I'm a hacker or someone who's trying to attack the Ethereum network and I try to go after the Go client, the go client maybe represents 60% of the network at

this point, maybe a little more, maybe a little less. If I find a vulnerability that's catastrophic that takes down all the geth clients on the system, then there's still the Rust parity clients that will still be running, because you would have to simultaneously find a massive critical bug in that.

**[0:24:49.7] JM:** I understand why you have testing across the different clients. My question was even more naïve. It was; why are there even multiple clients? Why do you need a Rust client and a Go client for Ethereum?

**[0:25:01.4] HJ:** Basically, if there is an attack on the network and it takes down one client, the others will be up. If you attack Bitcoin, you're attacking the Bitcoin core client, and that goes down. That takes down pretty much all the nodes that are using that specification.

**[0:25:18.6] JM:** Fascinating. You literally have different clients in different languages because it is a security risk to have everything in the same language?

**[0:25:28.2] HJ:** Yes. In our opinion, it would be a security risk to have the clients written with maybe not in the same language, but different full code basis of clients. So there can be multiple Java or C++ clients, but as long as there are different code bases that are just a copy of each other, then that makes it more resilient to attack in our opinion, and that's actually been proven out in a really interesting story we had back during DevCon 2 in 2016.

DevCon is our major annual Ethereum conference, and what happened was someone decided to attack the network by causing a lot of spam transactions while we were at the conference. So, luckily, because we were at an Ethereum conference, there was a lot of the Ethereum core developers there. So we got in a huddle and we determined that the spam transactions were slowing down the Google geth clients to a halt. So it was taking clients off the network incredibly fast.

We found out though that the parity Rust client was staying online. So if we had just have the Google Go geth client, the network would've been absolutely crippled beyond to a point where there would be a lot of vulnerabilities and chances for even further detriments to the network, but because the parity client was able to keep up, that saved us in a way and we were able to release a patch for geth and then everything was back to normal.

**[0:26:52.5] JM:** Okay. That's really interesting. So the state of Ethereum usage today is still quite immature, but there have been a few killer apps. You've had CryptoKitties and ICOs. I guess the ICO is arguably a killer app. Why haven't we seen a wider usage? Why haven't we seen more practical use cases for smart contracts?

**[0:27:18.7] HJ:** So the main reason is that the scalability and the technology is not keeping up with the demand. So where we would need a network that can support hundreds of thousands of simultaneous users were just not there yet. I mean, we technically can and the economics that would just cause the price to increase, but really what we're looking to do is to make this work for everybody and work for everybody without having them pay a ton of money. So I would say that scalability is one of the biggest factors that is leading to us not breaking out as a mainstream application.

**[0:27:59.0] JM:** So assuming scalability was solved, let's say we had infinite scalability on Ethereum, what are the applications that you think people would widely deploy and spin up?

**[0:28:09.0] HJ:** If scalability was solved, I don't think that there's going to really be a feature where people are on their phone and they load up Ethereum like they load up Facebook. I see Ethereum as the same thing as TCP/IP, which whenever you load up a web browser, you connect to the internet on your phone, you're using TCP/IP but you don't walk up to someone and say, "Hey, have you connect to TCP/IP today?" It's in the background.

So if you're technical, you know about it, you can manipulate the lower levels of the web, but otherwise it should be invisible and in the background. I think that's really how Ethereum is a breakout, like in 5 to 10 years I don't want to be hearing about people using Ethereum. I just want it to be an underlying layer that everyone uses without even knowing it.

**[0:28:55.6] JM:** Right. So maybe you interface with an application where there is some Ethereum smart contract that authenticates you and maybe there is an Ethereum smart contract that interfaces with an IPFS storage system that loads all your photos, for example. If we're talking about constructing the decentralized Facebook, maybe you have Ethereum mediating some of the computation of that decentralized Facebook and that you've got IPFS doing other

things. Maybe you've got Gollum doing other things, but just like we don't think of Facebook as this collection of TCP/IP and disk storage and in-memory databases. We think of it just as this big application. We won't think of the decentralized Facebook as specifically being on Ethereum. We'll ill just think of it is decentralized Facebook, but in actuality it will be a collection of Ethereum smart contracts and IPFS nodes and etc., etc.

**[0:29:59.3] HJ:** Absolutely. That's the future that I see with Ethereum. So as far as dapps or apps that are going to develop from that, the ones that are going to be most powerful are ones that dis-intermediate third parties, which is really the point of all this blockchain stuff in the first place, taking out the middle party or the arbiter that's dealing with your transaction. So, for instance, my startup did a decentralized, decentralized, dis-intermediated enterprise rent a car. So you load up and app, it looks just like a regular app that you would have if you were using a service like Turo to rent out your car, and you can add your car to the system. It would be GPS monitored using a he device in the car, and then using a hardware device in the car that connects to Ethereum and has its own public and private key. We send signed transactions to the car after reserving it using a smart contract and just putting the logic on the Ethereum chain and you're able to instantly lock and unlock the car and reserve it for cryptocurrency.

So stuff like that, 70% to 80% of it is invisible to the user. Right now you still have to have some stuff like cryptocurrency to run it and things, and this was only for demo purposes to be clear. Toyota hired us to do it, but it was still an interesting case of a real world use for blockchains to this dis-intermediate third- parties that would normally take a cut of running out your car.

**[0:31:22.2] JM:** Yeah. This is something I wanted to ask you about. I saw your presentations on this. This is your company, Oaken Innovations, and it is working on IoT security using the Ethereum blockchain. For many people I've talked to, they concatenated some buzzwords together and it is just buzzwords concatenated together, but you actually built technology. You've got some real stuff that you built, and I guess — So you said Toyota hired you to do that. What would be Toyota's perspective if they're hiring or they just like trying to explore this space and get a head start on how this technology might impact their business?

**[0:32:03.9] HJ:** Yeah. At the time they hired us for this in the early to mid-2016, they were working with [inaudible 0:32:09.6] such as Jim. What we did is we had different use cases that

we would do a demo for Toyota for and that they would bring it to their people to say, “Is this something that we want look at in the future?” That they see a future of autonomous vehicles. They see a future of different types of fleet management and things like the traditional car buying infrastructure and way you go and pick out a car being completely different. Why wait for services to dis-intermediate Toyota while they can dis-intermediate themselves in a way and still make a profit with these different models?

**[0:32:47.8] JM:** When you think about the idea of car sharing, utilizing the blockchain or being decentralized, this is a question that is worth thinking about right now or do you think it's just like a little too far off? To me, it seems very far off and it seems like we have no idea how Waymo and Lyft and all the kind of commercial changes to the car industry are going to impact the world, and we’re still very early with blockchain technology. Why is it worth it to even start thinking about the intersection of these two things?

**[0:33:25.1] HJ:** I think that it’s the right time to be looking at it from, like you said, a perspective of this is definitely a few years away. Uber and Lyft kind of came on to the scene and changed everything in just a matter of years, and I think that blockchain technology — The automotive industry has traditionally been an industry that slower to adopt standards, but with pressure from Tesla and others, they’re increasing their adoption rate for new technologies and things like self-driving cars and autonomous vehicles — Or, I guess — Yeah, self-driving cars that are autonomous vehicles.

So I think that it's the right time to look at it even if it seems a little bit early on. There's always other verticals and use cases that are more applicable today that you can focus on at the same time while still having this platform that you can build that will be ready when the time comes.

[SPONSOR MESSAGE]

**[0:34:21.5] JM:** You’re a successful developer and you couldn't have gotten to where you are without help in your education and career. Maybe you're thinking about ways to give back in the community where you live. The TEALS Program is looking for engineers from across the country to volunteer to teach computer science in high schools. Work with a computer science teacher in the classroom to bring development concepts to life through teamwork and

determination. Pay your success forward by volunteering with high school students in your area by encouraging them on the computer science path. You can make a difference. If you'd like to learn more about the Microsoft TEALS Program or submit your volunteer application, go to tealsk12.org/sedaily. That's tealsK 2.org/sedaily.

Thank you to the TEALS Program for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

**[0:35:27.8] JM:** I think that makes complete sense, and one thing that I think is hard to define from my point of view, and I would love to get your perspective on this, is how far there is to go in terms of getting Ethereum to a place where is scalable to a high volumes of transactions and where we will have consumer adoption? Because one thing that I think is interesting is that there were some days in the early internet where people made businesses like Webvan. Webvan was basically Instacart, but too early. The infrastructure wasn't there. People didn't have smart phones. It was a little too slow. That was arguably an era where people made businesses too early and those businesses were gated by real-world physical hardware infrastructure that was not invented yet. Whereas with Ethereum, it seems like the barriers to Ethereum being widely adopted, widely used are more in the software universe, and software advances so much faster than hardware propagation. Do you think that's accurate or is there something that I am missing here? Maybe like the consumer adoption of a new type of currency, maybe that is as difficult as a hardware adoption.

**[0:36:43.7] HJ:** Yeah, I completely agree with you for problem with the adoption aspect. There's definitely a problem with getting adoption and familiarity with handling cryptocurrencies.

**[0:36:56.0] JM:** It's very different than the .com era where you just simply did not have smart phones. You didn't have all these hardware that needed to be in place.

**[0:37:01.9] HJ:** I completely agree. The three biggest problems facing Ethereum this year in 2018 are scalability, UIUX and governance, in my opinion. So pretty much how can fit all these transactions and the popularity of our system growing. How do we make it easier to use, because right now it's very hard to use. Then going forward, how do we take what we have and

make it fair and a smooth process in the future when making very important decisions about the direction of the platform?

**[0:37:36.1] JM:** Okay, let's go deeper on governance. Ethereum is a huge open source project. Give me a high-level overview for how the Ethereum project is governed.

**[0:37:45.5] HJ:** Sure. So I don't consider any project right now having true “decentralized governance” completely, that there's always going to be, at least for now, some level of centralization. There're a lot of different strategies that go on to figure out how to run your blockchain or a decentralized software project.

The case of Ethereum, what we have is a bunch of people who are leading software contributors, leading thinkers in the ecosystem, people who've been around a while and some people who haven't been around a while who contribute to the decisions. That's the broad answer. The specific answer would be that we have the Ethereum improvement proposals in place, and whenever a change needs to be made to the protocol level of the Ethereum software, so for instance, things that require hard forks. So things like sharding scalability improvements in the future, or maybe upgrades to the virtual machines so you can include privacy or some anonymous smart contract security stuff with zkSNARKs. Whenever that needs to happen, people submit a proposal that has to be technically written to include things like abstract, a design mechanism, some test cases, things like that, and then every two weeks at the core dev meeting, if you've written in the IP or found in EIP that you think is good for the network and that needs to be implemented, you can champion that and say, “I want to add this to the next hard fork, or maybe it's something that we need to agree on on a network level that isn’t like to be added at a hard forks, but we do need to add it to all the clients.” So we just kind of have rough consensus on some things and then agreed to implement those changes.

At a community level, we also have a process for community standards. The most popular one being the ERC20 token, and how that came about was Vitalik Buterin and Fabian Vogelsteller — I think you actually had Fabian on your show recently, so he probably went over some of the ERC20 information.

**[0:39:54.1] JM:** No. We didn't talk about that.

**[0:39:55.0] HJ:** Oh! Interesting.

**[0:39:55.8] JM:** He was a great guest though.

**[0:39:57.4] HJ:** Yeah. So, basically, they came up with this idea for a token standard on Ethereum, and people started implementing it and using it and refining the standard. Eventually what happened was when enough people were using it, as the EIP became more mature, we decided to accept it, we being the editors.

Right now, it's kind of a an interesting time for Ethereum governance, because the editors have a lot of power it seems, so do the core developers and where we need to be is having more organizations, more diverse people and more entities, I guess, not even just actual companies, but entities formed to help foster this governance.

So recently there's been a group formed called the Fellowship of Ethereum Magicians, and their purpose is to kind of — Yeah, exactly. It's a little bit of a silly name. We pride ourselves on not taking ourselves too seriously within the Ethereum community.

**[0:40:56.0] JM:** By the way, that is what I love about Ethereum.

**[0:40:58.3] HJ:** Yeah. I mean, if you've seen, we're all about silliness and douche and these other stuff. So, basically, the fellowship is trying to be similar to the IETF. So they're not trying to make decisions —

**[0:41:12.5] JM:** That's the Internet Engineering Task Force.

**[0:41:14.5] HJ:** Yup, exactly. So the people who try to create — Who not try to. Who create and process the internet standards that everybody adopts. They're heir trying to kind of be that for Ethereum. So instead of taking the approach of like pushing an agenda or trying to be a political group, they're trying to be more of a technical working group, a technical group to make sure that there is technical soundness within the Ethereum protocol that things like EIPs that go in to the system are workable and going in the right direction.

I think that they're going to be a really powerful force in the Ethereum community along with EIP editors, along with core developers, along with the Twitter, Reddit, etc., community that kind of gives feedback. So the culmination of all of these groups creates the governance that we have today.

**[0:42:10.7] JM:** One point I'll make. I know this was an aside, the point about not taking yourself too seriously and douche and everything. I think that's actually pretty important. So I find Vitalik to be a really inspiring and magnetic leader, and I think actually one of the things I really like about him is he has a really good sense of humor. He's kind of hilarious. You even see this in his slides. He's just got absurd imagery, like you said, douche everywhere and rainbows and unicorns, which is weird, very hilarious things. Not taking yourself seriously is quite a deviation from the conventional open source software modus operandi. Like you think about the Linux community or the Bitcoin community, that the Bitcoin community seems like a lot of the — People from the Linux community migrated over to the Bitcoin community, and you see a certain seriousness. There's certainly a place for that, but I like the fact that Ethereum can make a lot of progress and serious scientific progress, like you've got very complex proofs that are being illustrated in whitepapers that are being written, but in the interaction with like at conferences or layperson interactions or the conversations I've had with people just on the podcast. There is a notion of fancifulness and like, "This is pretty fun. Let's just have fun with this." I find it a very unique culture and I really like it.

**[0:43:45.1] HJ:** Yeah, absolutely. I completely agree, and I think that there's definitely a parallel between not taking yourself too seriously and practicality. I think people get too tied up in dogma and their firm beliefs that they start to isolate themselves in these silos of communities and within themselves to not be open to new viewpoints. So not taking yourself too seriously is actually a very important part of the Ethereum culture and definitely lines up with trying to make something that the real world can use, understanding and admitting failures and trade-offs that we have to have in order to get the stuff out and usable by the average person.

**[0:44:27.4] JM:** And that pragmatism, by the way. You see that in something like the DAO hard fork, where the sort of conservative notion would be like, "No. This hack occurred and we're going to accept the hack for what it is," and then the result of most of the Ethereum and

community was like, “No. Let's just do a hard fork and iron this out,” and I feel like that was a notion of pragmatism. It was like, “We don't know everything. We made a mistake, and this happened, and let's fix it with a hard fork.” I like that flexibility.

**[0:44:57.3] HJ:** I agree and I'm definitely a part of the camp that likes that flexibility as well, even if back then I really went back-and-forth myself on whether or not the DAO folk should happen, which, by the way, that was the first week I was working at the Ethereum Foundation. So it was quite the welcome. The hack happened — Yeah, the first week, and it's going to always be kind of up for debate if it was a good idea or not, but I think that in the end it's kind of turning out for the better, the fact that we can show flexibility like that and also show that just because it happened once, doesn't mean that it's going to happen all the time.

I think Andreas Antonopoulos kind of likened it too, a small baby, like a two-year-old crawling and hitting its head. You accept that as like learning and growing when you're younger, but then as you mature and get older, you’re not going to hit your head all the time. Even if you do, you’re not going to have the same type of support that you would have if you were a baby. You just deal with it yourself.

So I think that's what's happening with Ethereum right now. I don't foresee us having a situation like the DAO where we revert, or not even revert a transaction. It was actually just in a regular state change, but that's more pedantic technical stuff. Either way, we won't have a situation like the DAO again for a while where we’re going to have a major blockchain revision that I can see.

**[0:46:16.9] JM:** I recall a couple weeks ago, I was seeing in my Twitter feed some debate around the Ethereum improvement proposal process. So the way that these proposals are made to change the spec for Ethereum. What are the debates around this process? What do people have subjective differences of opinion about?

**[0:46:39.7] HJ:** So one of the reasons that this came to light recently was the fact that we had an EIP brought forth by a developer from the, I think, Musiconomi platform. They were one of the ones who lost a large amount of ether and a failed smart contract transaction that was due to a bug in a smart contract. So, basically, someone wrote a smart contract, their group used that smart contract. It was a parody multisig wallet and the funds were lost, or when I say lost,

they’re not actually lost, they’re just stuck. The contract was built in a way where when this bug would happen, the funds won’t be able to move anywhere. So they’ve lost control of the funds. They don't have them anymore and they’re stuck.

So we could technically do a DAO style thing where the funds can be sent back to them, and so they did an EIP that basically just said, “Not necessarily for our specific fund reversal, but we want to make a process called the Ethereum recovery proposals.” My belief is it's ERP. I think that that's what it stands for, and it would be a standardized process for evaluating and recovering ether from stuck contracts or from mathematically provable lost ether.

There're been other bugs in the past and people misusing or mistyping smart contracts and addresses when sending their coins, that in a way can be provable mathematically. So there's been debate back-and-forth on if we should revert those transactions, and in this way they were wanting to do an EIP that said, “Let's make a process for this,” and the EIP was immediately met with a lot of resistance and a lot of people saying, “We don't want this to happen. It would be like another DAO and we’re too far along. As an editor, I even chimed in and said, “I mean, it's up to the community. This is an EIP, but for right now I'm personally not in favor of it, but at the same time, because [inaudible 0:48:41.1] an EIP editor, I need to also kind of wear the hat of not being bias and having a technical perspective on it to say, “This can be accepted if the community accepts it.”

The problem was the EIP process was so vague to so many people that there was just a lot of confusion with EIP editors not wanting to merge it and merge it into our repository, or not understanding if they should merge it into the repository. So what's happening with that is we’re doing a review and revision of the EIP process and to better define the roles of what an editor and a community member and an EIP writer and what that all means.

So, in short, it was a controversy over someone who is wanting to create a process to recover stuck or lost ether, and the community kind of gave them a lot of backlash, which caused a lot of drama and caused a lot of drama around the EIP process as a result.

 

**[0:49:42.4] JM:** Now is because the community, the core dev community did not want a formalized standardized process by which people would be able to recover the funds in the type of a situation like the parody multisig hack or the DAO.

**[0:49:57.7] HJ:** I wouldn't say the core dev community completely. I think it was mixed. I would say that the community response, the people who are the loudest were against it, but it's a very difficult and nuanced practice to actually cut through the crap and listen to everybody and try to figure out who's actually the majority or which one is coming closer to rough consensus on an issue, just because usually the ones who are against something are the loudest and you have to see what was it like outside groups or other maximalist from other points just [inaudible 0:50:31.3] the community and there's just a lot of different aspects of it that you have to kind of filter through.

**[0:50:37.1] JM:** What was the outcome of that, the parody multisig stuff? Did those people just lose their money?

**[0:50:41.1] HJ:** Yeah. In effect at this time, the funds from the parody multisig event are lost, and then when I say lost, I mean stuck. The contract had what's called a self-destruct — Or not even a self-destruct. It had a function where you could initialize a secondary contract that connected to the first one and someone found this out and went in and click the self-destruct button on the contract. So funds that were supposed to be processed through that contract no longer be processed through that contract, so in effect they're stuck.

**[0:51:14.5] JM:** Okay. I guess is the difference between the DAO hack and the parody multisig hack, is that basically boil down to an issue of scale, like how much money was lost? Is that how you would think about it?

**[0:51:27.5] HJ:** To be clear, there was to parody multisig incidents. The first parody multisig incident was last summer, in 2017, and that was an actual parody. They call it the parody multisig hack. It was someone who found a vulnerability within the parody multisig smart contract and stole a certain amount of ether and then a group called, the White Hat Group, went in and rescued the other ether that was in vulnerable contracts. Doing that saved hundreds of millions of dollars' worth of ether at the time.

Now, with the latest one that happened, I believe in November around the time of DevCon 3, November 2017, that was not a hack. That was a bug in the code that someone exploited. I wouldn't call it a hack, because they didn't actually gain anything from it. They went in just destroyed the contract. So it's just — They could've gone and actually done damage to steal coins, I believe, but they didn't. So that was an interesting thing.

**[0:52:28.1] JM:** The DAO hack resulted in a hard fork to basically revert history, and you could've made an argument for this happening I believe with at least one of these parody situations where somebody ended up losing money because of either a bug in the code or because of a hack, and I'm just wondering why those were not reverted in the same way the DAO was reverted.

**[0:52:54.6] HJ:** At the time that the DAO happened, it was roughly 14% of the ether in supply was used in this. Pretty much everybody in the ecosystem was using their ether to support the DAO in one way or another, invest in it or deal with DAO tokens. So it was the only killer app at the time. It was, really, the only app that was widely used at the time, because it was so early in the process. Now, there're so many different applications with different varying levels of fund loss. It's such a wider ecosystem now.

So it seems like something back then that have that much money and tokens at stake would have a higher impact, in my opinion at least, on the future of the very early nascent cryptocurrency and platform than it would today if they don't get reverted, which I mean it kindof showed, because at least from myself looking at the markets on the day of both the first parody multisig hack and the second one, the second parody multisg incident, the markets didn't really react to it. So it kind of shows that people don't follow the technical stuff as much anymore, and when these things happen, it's just the code acting as designed.

**[0:54:11.6] JM:** Well, Hudson, I know we're at the end of our time. So I want to thank you for coming on the show. It's been really great talking to you. The time flew by.

**[0:54:18.2] HJ:** Yeah, it's been great talking to you as well. Thank you so much for having me.

[END OF INTERVIEW]

**[0:54:24.8] 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 a 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 off-line. 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 ebook by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his ebook is about some of the distributed systems design lessons that he has learned building Kubernetes. That ebook is available at aka.ms/sedaily.

[END]