EPISODE 550

[INTRODUCTION]

[0:00:00.9] JM: We sign many different types of contracts throughout our lives. We sign a mortgage to get a loan for a house, when we go to the hospital we sign a piece of paper that defines how our medical data can be shared between organizations, these pieces of paper or PDF files represent our opting in to an agreement that will be mediated and enforced by computer interactions.

We can't see the code behind those computer interactions and we can't verify that it is abiding by the contract that we agreed to. Smart contracts allow for programmatic execution of contractual agreements, code is law and there is less ambiguity.

The most widely used smart contract platform is the Ethereum blockchain but several large enterprises are creating their own contracts. Should all smart contracts be decentralized or do enterprise consortium blockchains make sense? It's an open question.

In this episode, Marley Grey from Microsoft joins the show to discuss enterprise smart contracts and why you would want to use them and how they can be architected. Marley has worked on banking and financial technology for over a decade and he makes some strong arguments for why banks will adopt smart contracts and other financial systems too into their supply chain networks and the timeline for how that might take place is also discussed.

I know that some people are a little tired of the crypto currency episodes, so we would love it if you filled out our listener survey at softwareengineeringdaily.com/survey, so we have some concrete feedback or you can send me an email jeff@softwaredaily.com, I would love to hear from you. You can also join our Slack to interact with our community at softwareengineeringdaily.com/slack and in the meantime, of these cryptocurrency episodes, if you're really tired of them, just check out our back catalog of episodes at software daily.com.

Where you can download our apps which have all of our episodes including our greatest hits list. The apps will soon have features like offline downloads and bookmarking, so it will be more like a legitimate podcast player. I hope you enjoy this episode.

[SPONSOR MESSAGE]

[0:02:19.6] JM: Today's sponsor is Data Dog a cloud scale monitoring and analytics platform. Data Dog integrates with more than 200 technologies so you can gain deep visibility into every layer of your stack and any other data that you're interested in tracking as well.

For example, you can use Data Dog's restful API to collect custom metrics from your favorite crypto data sources and analyze trends in Ethereum prices over time. Start a 14-day free trial and as a bonus, Data Dog will send you a free T shirt. You can go to softwareengineeringdaily.com/datadog to get that free T-shirt and thank you to Data Dog for being a continued sponsor.

Get that free T shirt at softwareengineeringdaily.comdDatadog.

[INTERVIEW]

[0:03:17.3] JM: Marley Grey is the director of the enterprise Ethereum Alliance and the principle architect at Microsoft, Marley, welcome to Software Engineering Daily.

[0:03:24.3] MG: Thank you.

[0:03:26.0] JM: Before you started working on blockchain technology, you were working on technology strategy for banking and capital markets, financial systems, you were working on these things at Microsoft. What caused you to shift over to focusing on decentralized technology?

[0:03:43.8] MG: Well, it was the introduction of concept of a smart contract which unlocks sort of a lot of other potentials and second phase was the introduction of having private blockchains, where we could get over some of the regulatory obstacles that prevented blockchain from being

sort of a viable solution for – to be used in that industry. That was the thing that really unlocked it and then that sort of opened the Pandora's box if you will, from that point forward.

[0:04:15.3] JM: Back in 2014, 2015 or around the time Ethereum came out, what was Microsoft's perspective on decentralized technology?

[0:04:24.8] MG: We were early embracers of Bitcoin from accepting it as a payment instrument for our X Box and market place and things like that just to see what it was going to be like. From a decentralized point, we have a long legacy of decentralized technologies around, if you go back to our groove, sort of a decentralized sharing of files.

There's a whole bunch of things in the gaming arena for multi-party games and then we experimented a lot with the combination of the cloud with decentralized technologies and what the capabilities there would be. We've done a lot of things sort of across the board of Microsoft in decentralized technologies and so I would say, there was a healthy understanding and also a respect of some of the pitfalls that have occurred in the past.

[0:05:19.7] JM: I think that there is this tremendous wealth of opportunity that's going to arise as we see the blockchain cloud solutions emerge, it definitely feels like it's very early – how has your thinking around the ways that blockchains can interact with a cloud provider? How has that thinking evolved in the last couple of years?

[0:05:43.7] MG: It just keeps refining. One of the first things I went about was to immediately try to think about, what were the limitations of a decentralized sort of smart contract based system and it was always around the scale and build and execute the number concurrent contracts to be able to get to sort of internet scale which you know, I immediately knew that we would need to introduce something like separation of concerns, to be able to separate the logic execution from the actual ledger itself.

But then, somehow tie them together cryptographically so you preserve the integrity of the blockchain and when you look at the historical record, you can not only trust this integrity but you can also build upon historical truth and just keep going there. That has evolved, how would

we do that? The evolution of this has been really finding the technology's solutions and at the time, certain things didn't exist.

The notion of what we could do from a privacy perspective for running logic in the cloud, using something like an enclave. Intel's SGX chip became very interesting, solved a particular problem for us and then the whole multiparty aspect of the blockchain was decentralized but we want to be able to have multi party shared multi party compute.

The other thing we wanted to avoid was to prevent duplication of effort just for the sake of duplication of effort to slow you down. If we can figure out a way to increase the performance and the scale by again, preserving that integrity, improving that integrity. So that the evolution of that has been, how do we do that, what are the frameworks that we do that. What's our first meta-market, we've come a long way and we hit lots of dead ends and had to back up and go other directions. In general, the thinking is how do we make the cloud evolve in a decentralized network better together?

So w let each optimize what they're good at doing and not try to do everything in one place, that's the ode of failed architectures in the long run, when you're trying to get to scale that we're talking about for public blockchains or even consortium blockchains.

[0:08:02.1] JM: You've written about this unease that you had early on when you were looking at smart contracts and the way that smart contract execution on Ethereum works is that when you write a smart contract and you deploy it to Ethereum, that smart contract gets deployed to every Ethereum full node.

When you execute a smart contract, that execution takes place on every Ethereum full node. That's a great proof of concept way of having decentralized computational consensus but it's much less clear that that is the end state of how we do decentralized execution of computer programs because something instinctually feels less efficient about it.

Even if you look at the decentralized efforts to make these smart contract execution more efficient like you look at something like – I think Plasma does this where they have side chains

that are – you know, the computation is not necessarily preceding on the main Ethereum chain but it checks in with the Ethereum main chain occasionally.

Anyway, you're starting to see these host of different models for pushing side chain computation. You know, you could imagine that the same sort of thing with the cloud provider with you know, having these trusted consortiums that trust each other, doing their own side chain computation, maybe they check in with the main chain, maybe not. But there's obviously a variety of ways that this computation could take place.

It doesn't surprise me that there has been some sputters and false starts because this stuff is really hard.

[0:09:53.5] MG: You know, if you look at the way the initial implementation of more contracts is done, it was done with the assumption that everything had to be completely trustless. In that manner, running fully deterministic code and having every node execute that contract and to achieve the consensus that – of the results.

Doing that in serial, so you can't do it in parallel, it basically gives you that consistent state and lets you have this achieved consensus. That ultimately, that one assumption isn't always the case. For the most part, in the enterprise, is that will never be the case. If we look at the interest in blockchains and across industries where you have strong regulatory constraints around knowing your customer.

You can't have sort of trust issues. You are going to know who the counter parties are in your contracts and once you start to do that, then you start to say, "Well, really, the only parties that are going to care about the result of this contract are not everyone in the network is going to really care, it's just the parties that are signed up and perhaps a regulator."

Then, how do we prove that to the satisfaction of all those interested parties? That let's you then say, "Okay, well then, what are the assumptions we can have?" A.) Is e would have some sort of backing of strong identity and then B.) Is we know the public cloud is going to be, whether it's Azure or AWS or you know, Google, you can always have some cheap, elastic computing out

there in the cloud that you should be able to reach out at run time and spin up a secure execution module to run some logic for a contract on this particular blockchain or blockchains.

Then you can start to do all sorts of interesting things when you do that. That lets you start to use to do interoperability between chains using middle tier rather than trying to integrate chains directly node to node which is again, if you go back in the historical context, integration efforst at the data tier are just riddled with disastrous results.

It's much more efficient if you have a lot more nimble integration capabilities when you sort of do it in a sort of a brokered fashion with a middle tier component that gets taught to each different platform and make reasonable choices and pragmatic with decision points that you're not always going to get and be able to adjust the changes and not be such a brittle integration.

When you start to look at these things and you could say, "These are the assumptions that we can go with, we have lots of design choices that we can make." The simple and ones that whole – how we just in time go out and grab some secure container to run some contract-based code and then we can prove that that contract code for a series of proof and not just one single proof but a collection of proofs that will satisfy the requirements of the most.

Well, not the most but your 90% of the market, would be satisfied. That it was done only up and up and it's far with what they're doing today on pen and paper and fax machines and you know, all sorts of processes that are out there that this could replace.

[0:13:08.8] JM: A few use cases that I consistently hear around the discussions of enterprise, consortium, blockchains are one, you've got the question of alone so Marley, you loan me some money and we want to have that loan codified in a smart contract and maybe we don't necessarily need the entire blockchain to verify the transactions and the payments and the payment schedule of that smart contract.

Maybe we're fine just leaving it to some set of nodes or maybe it's my bank and your bank and they're maintaining consensus with each other through some set of nodes. There's that kind of use case. There's also the use case of the supply chain where you know, you use smart

contracts to verify that as for example, an apple makes it from an orchard, to somebody that's holding that apple in a warehouse, to somebody that is selling that apple at a grocery store.

You know, that apple, it is the same organic apple that was grown at the orchard and it was not swapped with an inorganic apple somewhere along the supply chain. You could imagine using these consortium blockchains for the same sort of purpose. Again, you don't necessarily need the entire Ethereum blockchain to agree that that apple is still the same organic apple, maybe you just need all the different players along that supply chain.

You need the person who grows the apple, the person who stores the apple for a while, the person who sells the apple and maybe even the person who buys the apple. You just need this subset of people who are involved to agree on the transactionality. Would you agree that those are both kind of use cases that we could be thinking about from the consortium blockchain level?

[0:15:08.1] MG: Yes, these are the beginning use cases and we'll see them get more and more advanced as people start to get a hold of some of this next generation, multi-party compute and distributed sort of application model, we start to see these use cases sort of to get more and more complexed and a whole new businesses will be created in the sort of ecosystem.

These are some of the ones that most people can sort of reason and understand to begin with.

[0:15:39.2] JM: Right. Just to play the devil's advocate here, because I think that these enterprise consortium blockchains make sense but just to play the devil's advocate, for people who I know are skeptical of the enterprise blockchain stuff.

Why would you need a blockchain for this? Why wouldn't you just have a database with shared permissions or even if you wanted to get crazier, you could have a virtual machine that all the different players involved have access to, a complete log of the virtual machine.

Maybe the virtual machine has a verified open source operating system, so everybody can agree on the operating system, everybody can agree on the programs that are running in it.

Then maybe has a very detailed log of the computations that take place in that computer, why do you need a blockchain to fulfill those kinds of things?

[0:16:31.3] MG: I usually try to put it in a simple form and say, let's assume that all history of mankind is recorded in one place. We all go to that one place and it's one physical location, say a library, central library of the world, or humanity or whatever.

Those actions happen, events happen, they get recorded there and then maybe some atrocity happened and somebody wanted to go erase that from history and do that in a way, even though all the data from that point forward is built upon those things, becomes a very simple exercise of going into that one source, manipulating the data and erasing some data and then rebuilding the cryptographic proof from that point forward all the way to the front.

There's no one to dispute that, there's not another copy of that data that can dispute that historical change, somebody has gone back and changed history to that effect. When a blockchain essentially says, "Well, not going to have just one library, we're going to have thousands of libraries" and to be able to do that and the libraries are constantly in sync and they have a sync window of say, make it an hour, you know? For someone to be able to erase history, they would have to be able to perform that function and that step in a coordinated way within that time window of all the copies at the same time.

To pull that off which in itself, it's just one sort of scenario where you know, having a blockchain or a redistributed set of distributed truth is a powerful one and it creates this sort of, the foundation for trust, and even trustless environments that we get there. That's probably the first step from just a recording and the ledger and keeping a ledger distributed and what the benefits of that and then you could throw in the disaster recovery.

You know, the place physically where the database was stored, was blown up or something, and its backup was in the next region, you know, it was a nuclear attack or something, you lost your data where as if it's fully distributed, you probably have copies around the world and maybe on the space station, who knows. I mean, that's one piece there.

[SPONSOR MESSAGE]

[0:18:54.9] 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, although 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 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 going to find out if SE Daily listeners are above average.

If you're looking for a job, Triple Bite is a great place to start your search, fast tracking you at hundreds of top tech companies, Triple Byte takes engineers seriously and does not waste their time. I recommend checking it out at triplebyte.com/sedaily. Thank you Triple Byte for being a sponsor.

[INTERVIEW CONTINUED]

[0:20:25.7] JM: If I think about that explanation, I think what I hear you saying is, we could do this, we could have these shared consortium blockchains built out of our classical databases and blob storage systems and logging systems and operating systems, we could do that. I think what has happened is, Bitcoin came out, Ethereum came out, and people started thinking deeply about these ideas of trust and you know, Bitcoin and Ethereum illustrated the extremes of trustless environments, although, it's still autonomous so maybe it's not completely trustless, it's still centralized in terms with the proof of work. Miners being centralized but it's a pretty trustless decentralized system.

At least just as a thought exercise. Then, what happened was, people started thinking about "Hey, why haven't we been thinking about this way that you know, maybe subsets of the entire world can share a source of truth." Like you know, maybe you want to have banks to have a central source of truth that just those five or six banks are referring to or maybe you just want

these, you know, these different people in the supply chain to be able to agree on this space of data in this history of transactions.

Again, it's not necessarily that these shared trust systems, that are not totally none trusted, it could not have existed before blockchain and proof of work, it's more that the zeitgeist, what people were thinking about got changed by Ethereum and Bitcoin and so a company like Microsoft started thinking, "Wow, this is pretty wildly applicable, how could we build that?"

You don't even really need like a proof of work system. Its' just more like, "This is an insight that we could apply to different business scenarios that we're working with." Am I articulating it correctly?

[0:22:28.6] MG: Yeah, the common fallacy people – they approach this and they think of, well trust is bullying. It's not. Trust is not a true or false, there are ranges of trust and I usually have people who say, "Okay, let's assume that you just got your brand new Porsche, you drive it off the lot and you have two people coming up to you and asking to borrow your keys, they have to go somewhere in an emergency, all right? If it's your spouse or your brother, you're more apt to give them the keys."

"If it's a total stranger, you're not going to give them the keys, right?" Maybe it's someone that you work that you know from work. Just in passing. You would say, "Hey" to them, but you're not going to give them your keys, your brand new treasured possession because even though you might say you trust them but you don't trust them like that, its' a sliding scale and then the – That's one of the things that people walk into some baggage when they think about blockchain as they say "Hey, it's either trustless or it's fully trusted."

That's not – the word trust is analog. And then the second piece is they entangle the truth in the resolution of the truth together and that's where, when I started, I would try and ply those two a part and say, the blockchains where we persist the truth, where we store historical truth and current state of the truth.

But we resolve it somewhere else and that resolution of the truth, we can optimize for whatever the use case is and the only thing we have to do there is just have the proofs of where we're resolving the truth and persist those with the truth.

Along with the truth so that we can have the ledger to see the truth and then have the proofs of how it was resolved. That really gives us a sort of elegant sort of approach to sort of building systems on these two principles.

[0:24:20.0] JM: Maybe we could just talk about the enterprise architecture of a smart contract. We've done several shows about how smart contracts work on the Ethereum blockchain. How would you contrast that with how smart contracts might work within an enterprise consortium? A group of enterprises that want to operate in a way where they don't necessarily trust each other, so they just want to have some way to have a system of trust between those enterprises?

[0:24:54.1] MG: Right, the way we've architected this for what we're calling them enterprise smart contracts and used to call triplets from developer standpoint but essentially, you still use a ledger to define – every contract has a couple of properties, one of them is essentially have some sort of business schema. How do you define what it is your contracts are keeping up with?

Usually, it's the terms, who the parties are, if it's a loan, it's how much are you borrowing, what's the interest rate, what's the payment schedule, when is the payment due, what happens if the payment's late. You sort of rapport all of those things and then people agree on and they sign on and that thing should essentially execute. That largely stays the same.

You're still going to record essentially the schema and then all of the transactions as that thing becomes bounced, each of the payments and the recording of those payments are then recorded and that's more a contract. The difference is, the logic, once it's sort of bound and executing can then be pulled off chain and there will be a cryptographic set of trust. So like in Ethereum, you can use what we call function modify, which can essentially guarantee that only certain parties can alter the state.

We assign that some code that's running, that the parties have agreed to. We can then have that logic written in a language, java.net, whatever. Just figure some run time that you want to be able to support. The counter parties can give the brand new contract, you'd have some vetting to go through but let's assume it's one that's been tried and true, it has a great reputation, it's been thoroughly threat modeled, all that good stuff.

We could just pick it out of a market place, right? It's a loan, I cripple it, we can then say "Hey, this guy is going to control our loan and we agree on that" and it's going to generate and execute and the results of this loan and it will persist only the result of its calculations, it records the truth of the payments and you know, if there was a pin of the – how is that determined but also the proof that go along with it.

That essentially lets us pull that logic out and run it, a set of code that's running in the – sort of a shared model. Then the code runs once essentially and we can run it in a shared environment like the cloud. In this case, if you don't do it like that, usually what will happen is, when referring in duplicate, each of us is responsible for bearing 100% of the cost of the contract but only deriving maybe 50% of the value.

If we run a shared environment, then we can split the cost or – equal to the amount of value that we receive, or I can also use the cost of computing that result as an incentive. If I'm the seller of a contract and you're the buyer, right? As an incentive, I can tell you, "I'm picking up your fees." Does that make sense?

[0:27:52.8] JM: Yeah, it does make sense. You mentioned along that definition, the term cryplet. Can you define what the word cryplet is?

[0:28:00.7] MG: Okay, yeah so cryptlet is the term that I came up with way back in 2015. It is just essentially a play on words. So the servelet is the middle ware component that's a piece of code that runs and understands and understands the crypto graphic premise that blockchains do but it also has the ability to create, to do crypto graphic operations, without the developer having to know that that's what they are doing.

So it is essentially you write your code in whatever language that you want to and focus on the business logic of these different things and to be able to have that result that you have computed in your code automatically be proven crypto graphically. So digital signature is created, you can do things like encryption, you can do all sorts of things and not have to write the code. So the cryptlet is sort of this platform that you write your code in.

That wraps all of these capabilities and then all subtract your business logical way from the actual blockchain itself. So it will work across any type of blockchain, very similar to most middle ware if you go back and look at J2EE and database abstraction between Oracle related shown database and that sequel server and then formats and then that really freed up and unlocked a lot of solutions to be built in a cross platform and across database way.

We are talking about the same thing for blockchain. So you can build a solution or an enterprise smart contract that it will work against. You know Ethereum, Quorum or High Pileger Fabric or Acorda or whatever, it becomes this portable contract that you could use. So that's what cryptlet is, it runs in the cryptlet framework and Asher and it's sort of a cloud. This is shared sort of execution space that a cryptlet runs in.

[0:29:50.6] JM: And so would you say that a cryptlet is – essentially it is a smart contract that adheres to the components, the definition of a smart contract that you defined in this whitepaper that you wrote. So you've got some different components, you've got the schema, you've got the logic, you've got the counter parties, you've got external sources and you've got the ledger and then the contract binding that brings it all together and the cryptlet is just the notion of a smart contract in an enterprise blockchain, is that right?

[0:30:23.8] MG: Yes, the accurate description is contracts and if you've ever taken in a contract whichever everyone has, everything is a contract essentially but usually the contracts are templates and no one really writes contracts from scratch. The longer you are pulling clauses and things together and composing them of these components and then it becomes an instance of a contract when the counter parties have signed to it and then just execute.

So we're talking about the same thing here, it's just we're decomposing those actors together and let them being composed in this framework and then the cryptlet is just the logic portion of

that. Of the blockchain, you can tell it too. If the framework has being configured to run across three different blockchains when you're instantiating an instance of it, you tell it which blockchain you want to have it home to and it essentially binds to that blockchain for its primary state.

You can start to assume very interesting things by having other cryptlets using other cryptlet setter pointing at different blockchains to create sort of smart contracts that work across simultaneously across different blockchains. For example, moving or trading assets that have been recorded in one blockchain and transferring them on a completely different blockchain on a whole other platform, you could execute the transaction like that. In fact, we share that in one of our examples and sample code for the cryptlet's framework today.

[0:31:51.0] JM: So I am wondering to what degree enterprises have started to adopt this stuff or if they are just starting to tinker with it and if there are some sort of bottle necks that they are waiting to be surmounted before they start coming into the space. Like what is the current state of enterprise adoption of these kinds of blockchains?

[0:32:15.5] MG: Well the investments are pretty intensive from a use case. We have gone from an evolution. Early on everyone was doing these hackathons across companies and trying to figure out what blockchain they were going to use, right? They would stand up in different blockchains and they would compare them against each other and they would deem one the winner and then they would go forward, which turns out to be completely the wrong way to go about it.

Because they didn't know what they were going to build on top of it, so you have no idea based on the requirements that you selected in the said blockchain of where they the white things that you were indexing for and it turns out you're wrong, you just spent a lot of time. So there's a couple of things that we have done to make that easier. So in Asher, one of the first things we did was try to make a blockchain that is ridiculously easy to stand up.

It was very difficult to stand up a private blockchain before we did that and now, you could do it in minutes and stand up in a fairly sophisticated hundred node geographically distributed blockchain for you to model things after and do it very quickly and cheaply. So that makes it

really easy. Then sort of, “Okay let’s figure out what we want to do first,” so that’s probably the first thing is to get people to stop trying to focus solely on “What’s the right blockchain?”

But figure out what is the type of application, what does it mean to be a multi partly application? And it turns out that is a difficult thing for people to get their head around and we see that and it is sort of a fortunate thing I say is it’s taking people a lot to figure that out because also it is taking a while for us to build the platform that is truly multi party, that works across multiple distributed letters that technology has to. We think it is keeping pace where customers are.

So our customers are prototyping and they have a private early adopter program going on right now and that’s in a certain degree sort of that obstacle has become unofficial. For us, as platform providers to be able to at least keep up with customer thinking and that’s been the big shift. It’s the total mind shift the way you think about things and early on there were a lot of mistakes and you know people building applications on blockchains, ahead of business being on blockchains. A centralized database is a way better type of solution.

[0:34:30.3] JM: I’m sorry, I didn’t understand it super well but is there a technical bottle neck to them adopting it or is it just a lack of understanding or they’re gun shy?

[0:34:42.0] MG: There’s a couple of them that are at the technical bottle necks. First is there’s very few people that have the experience. Let’s take Ethereum writing solidity. So there’s not a lot of people out there and there’s very few people that can say they’ve written production code for solidity and the definition for production is different for different things, right? So the enterprise is up, the production word is a lot different than it would be for say a public Ethereum smart contract.

But there are slews and armies of people that have built production systems in Java and C Sharp, in other languages, and there’s an army of patterns and practices and tools and it is all there. That ecosystem is there and so the obstacle people go in with and they try to do solidity and they’re like, “Where are the tools, how do I debug this thing?” It’s hard because the tools are not there. The documentations are not that easy to get a hold of.

So that in itself is a very large obstacle, knowing that there are not enough skills out there, but the tools are not there for people to really scale this up. So, I think this approach that we are building here is addressing that piece and then the other technical obstacle is what the blockchain tells I have to catch from performance, perspective, solve problems like privacy as well and then permissioning and things like that. Those are being worked on as well.

[SPONSOR BREAK]

[0:36:16.7] JM: Software Engineering Daily is brought to by ConsenSys. Do you think blockchain technology is only used for crypto currency? Think again. ConsenSys develops tools and infrastructure to enable a decentralized future built on Ethereum, the most advanced blockchain development platform.

ConsenSys has hundreds of Web3 developers that are building decentralized applications, focusing on world-changing ideas like creating a system for self-sovereign identity, managing supply chains, developing a more efficient electricity provider and much more.

Listeners, why continue to build the internet of today when you can build the internet of the future on the blockchain? ConsenSys is actively hiring talented software developers to help build the decentralized web. Learn more about consensus projects and open source jobs at consensys.net/sedaily. That's C-O-N-S-E-N-S-Y-S.net/sedaily. Consensys.net/sedaily. Thanks again ConsenSys.

[INTERVIEW CONTINUED]

[0:37:33.1] JM: So, if I understood you correctly, there's a couple of things that you are working on to improve the state of adoption and the difficulty of enterprises getting started with this stuff. One is you're building out tooling. So better ways to write prototype test and eventually deploy enterprise smart contracts, that's one part of it and then the other part of it is actually working on the runtime, the execution environment of the smart contracts.

So you've got both of those fronts that you need to make some progress on before you see for example, Goldman Sachs deploying a future's contract to enterprise blockchain.

[0:38:18.3] MG: Yeah but that work is happening now both the customers building out what would they future's contract look like on this type of enterprise blockchain. Then what our customers are seeing and what they are saying in this market is they can't wait for the technology to be fully baked before they try to understand what the new business model is and that's the fear that everyone has. When the enterprisers look at this and they see this disruptive technology it's not like –

It really is going to change business at the DNA level, so they can't wait for this all to be figured out. They have to be in there figuring out the business model because if they don't, they are going to end up like Blockbuster and they don't want to do that. So yeah, we are approaching it that way and we are also – the ledgers themselves, we're helping not directly, sort of indirectly, trust sort of the enterprise capabilities that blockchains and the characteristic they can have. They could have it and try to do some heavy lifting there as well to help that overall community.

[0:39:20.2] JM: It must be frightening and at the same time exciting because I am trying to report on all of this stuff and it feels to me like unless you are a core developer and you're really at the forefront and you have an understanding of all the research at a very fundamental level, everybody else is fumbling around in the dark trying to understand exactly what's happening and what timescale we are on. I think most people who spend just a little bit of time in this space understand that things are happening.
And this is going to have real impact on the way that we do business. But many fewer people truly understand the trajectory. It's more like this very misty landscape that we could barely see beyond the six-month ahead of us time horizon as to how the space is going to evolve and how it's going to impact things. I imagine it could be both exciting and a little bit disconcerting.

[0:40:21.1] MG: Oh yeah, it is and there's a lot of that fumbling around but it really has been like that for most technologies. The big driver here that I will say being in this industry for so long an just technology in general, from a technology standpoint was most technologist in IT departments around the world missed this completely but it was the line of business executives, the business people that saw this and they are the ones that are driving this.

And they just happen to be the ones that write checks and they have the budgets and a lot of these enterprises their IT department is a cost center, rather than a revenue generator and now you have the revenue generators throwing money at this thing saying, “We’ve got to figure this stuff out now because if we get it wrong we could go away,” you know? “We might be out of business” and so we have that whole fear and greed on that side really driving this.

So that’s why IT feels like, “Gosh I am running so fast and it is very foggy. I don’t know if I am about to drive off a cliff, or hit a pile on, or if I am going in the right direction even.” So yeah, totally like that but the good news is that it goes forward. That window of how much further you can see in the fog and the distance is getting further. So we are starting to shed some of that. We are getting into the phase where everyone is writing their own blockchain.

So we are settling on a couple of those and that will continue and you will start to see further and further and ahead and can make more and more informed decisions as you go forward from a technology and limitation standpoint.

[0:41:55.3] JM: So talking specifically about the process by which people write smart contracts and test Ethereum applications on Asher for example. What’s the workflow like for developers who are building smart contracts today and what specifically are some of those problems with the developer experience?

[0:42:21.3] MG: Well, you know the first thing you have to stand up on network. So that is a trivial exercise, you can deploy a blockchain network in a few minutes and then secondly is, “Okay, well how do I even write up smart contract and then how do I deploy it to the blockchain? and then once it’s deployed how do I interact with it?” And then you go out and look and there’s five different answers for that. So if you’re a dot net developer, you would use –

Go out and find a library like Ethereum, which is a great client for Ethereum networks. It makes it for any dot net developer pretty simple to take existing solidity code and create a C Sharp client or a VB client that knows how to create a contract and then can execute transactions against it pretty easily. You can do the same thing in Java Script with the Web 3JS and then there’s block IO’s go to Java implementation. So there’s a bunch of different ways for doing that.

But you have to say, “Okay who is the developer and how do they get started?” so that’s probably the thing that we would recommend. Now some developers will go in and say, “I want us to do everything in Ethereum. I’m going to write my solidity code and I’m going to write my own client and I’m going to do these different things.” And it becomes very difficult. Key management, where are you keeping your keys and those types of things.

It’s kind of cumbersome. So you have this whole infrastructure standup and then you have to figure out what are the libraries available for me to build this and let me go in and find some samples. The good news is there’s lots of samples out there and you can get everything from samples for dot net to Java to Node or whatever platform you go onto. So that’s the second one but then really the thing that you have to do is stand out, sort of something demonstrable that has a user interface that a business person can understand.

That second piece becomes very difficult as well because most of these technologies are built to be single party where it’s a single organizational website, right? It is a multiparty sort of experience. So it becomes very difficult challenge, so that workflow becomes difficult as you crawl at the stat because there’s less and less sort of samples to look at for how they do that and that is one of the areas we are looking at.

And where we are investing is to try to fill that sort of void that occurs as you step up and say, “Okay I’ve got the basics, I can create a smart contract and I can send transactions to it, I have a great command line interface, great now what do I do?” That’s what we are working on right now.

[0:45:00.0] JM: So if I think more concretely about a use case. Let’s say I am a Goldman Sachs type of industry and I want to offer a future’s contract that people can transact with. So, in this model I would – first of all, I am spinning up my own blockchain. I am creating a consortium blockchain and then I guess other people would opt into that consortium somehow or help me understand if I am Goldman Sachs and I want to make my own blockchain, why would I do that and what would be the process of onboarding other people who would opt into that blockchain and then eventually setting up something like a future’s contract in smart contract form?

[0:45:47.8] MG: Right, so you first have to see the network. So you can just say, "Let's use Ethereum," and so if I am customer I would then say, "I need to see that" now what does that mean? I am going to create a couple of nodes. I can do that in Asher if I wanted to do this, I would create essentially the first slice of the consortium pie. Then if I found some willing participant then I would invite them to join my consortium meeting.

And then what they would do is they would spin up some nodes and join that network that means that now my nodes and their nodes are talking on the same network. So transactions are flowing back and forth and then we do that for each subsequent party and let's just build the consortium network. Now those nodes can be anywhere, they can be in Asher. We make it easy where you can just create the first seed and then invite others and then they can go on Asher and they can say, "I want to join this one."

But in reality what people are going to do is they'll have their nodes and their own data center or some other cloud provider and sitting those together but essentially that's the first step as you've got to build the network. But you don't have to wait to build the future's contract until that network's but you can go and build it on your own private and then you can then move it to whatever blockchain once it gets established. So you could do these two things in parallel.

And that's what we're seeing now is people are starting to build apps for the consortium network is even finished and starting to build these solutions and doing them on parallel again because a lot of these businesses are really pushing the gaps here to get something done.

[0:47:18.6] JM: And what is the process by which – so if that future's contract gets created, how does it get deployed? Does it get deployed? I guess it gets deployed to each of those nodes like it gets deployed to my nodes and your nodes. If I am Goldman Sachs, I create the future's contract and you're somebody else who wants to use that smart contract, you spin up your nodes to join my blockchain and then we're both running the same smart contract. I guess we replicate our transactions just like the Ethereum full blockchain would, that's the system basically?

[0:47:51.7] MG: Yeah, the smart contract is again for instance, until you have created this it is still called the constructor on it and it would deploy. All you have to do is to send it to one node

and it will create that instance of a smart contract and then replicate all on the other net. So the state of that contract, the sixth address is there and then I can tell somebody else that wants to be a participant in my contract what the address of that is and then they can look at the terms and they could sign on to it.

And then once it becomes binding, we just look at that one address and we know that it's this type of contract. So we know what the scheme is and then we are just going to execute a logic and that's where you say, "Do I execute the logic in the solidity code on the smart contract on the chain or do I want to run that off in the cryptlet and what are my performance characters?" So in this case it's probably going to be something they want to have high performance.

So we can then say, "Okay we are going to run this in a cryptlet" and then we'll go, "Geez, where do we want to run it?" Well we need it in Switzerland and in this case, Asher Land, Asher can be his Switzerland and we're going to agree to run our cryptlet in Asher and it will write the results to this address and we can all see the results and see the proofs of how that contract executed from this code that we are sharing the cost for in Asher. So that's the workflow.

Now you don't have to obviously do that. You can run all the code in Solidity and you'll see the same thing. It's just your logical run on every single node on then network and it limits some of your privacy because we do it in cryptlet. We can encrypt the payload before it is written to the blockchain, all sorts of things that you have a lot of advantages in the cryptlet as architecture.

[0:49:31.0] JM: So I want to bring this point how because I know we're near the end of our time but I think this is what's really important for people to understand about this and you could tell me if I am wrong. But it seems like again, people could have, you could have built futures contracts on internet infrastructure before. I mean people have done that but in the past, I mean I am not an expert in financial services implementations, but my mental image of it is that it is not easy to understand the infrastructure.

It's kind of Clue G, the integrations are really troublesome and it takes a long time to do and this architecture of having – we each have some nodes and we replicate our contracts across those nodes and then we've got a programming language that defines how our transactions are going

to proceed. It's just a better programming model than the past craft that we've built up in the electronic financial system, is that accurate?

[0:50:29.5] MG: Yeah it is. The other way, the previous way we did that was we built it with settlement houses. So if you look at like the DTCC that they settle derivatives that are over the counter derivatives, they are actually a third party but each of the banks, the big brokerage houses they spun it up and they created a middle man, which creates a long settlement time and if you look at the equities market, it takes three days to settle the trade.

And if you actually want to have the assets that you sold it for, liquid, it takes only two days for it to be deposited in your checking account where you can't do anything with it. That five days sort of latency it's all because we're settling on a centralized database that then has to be reconciled with each broker's ledger So we have duplicated ledgers across the brokers that were involved in the trade in this once central clearing house and the overhead comes in the fact that we have to then validate the ledger across each one of those.

So that takes time, we are all settling to one single ledger, we can settle in near real time which them removes that sort of friction and cost and inefficiency in the network itself.

[0:51:40.5] JM: Okay Marley, well I know we are at the end of our time but I feel like I understand this space a lot better. Thanks to this conversation. So I appreciate you coming on the show.

[0:51:47.7] MG: Sure, great to be here. Thank you.

[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 is great to see the continued progress on GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily and 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.

[END]