Catalyzing Computing Podcast Episode 6 - Interview with Keith Marzullo Part 1

Intro [00:00:10]

Khari: Hello, I'm your host, Khari Douglas, and welcome to Catalyzing Computing, the official podcast of the Computing Community Consortium. The Computing Community Consortium, or CCC for short, is a programmatic committee of the Computing Research Association. The mission of the CCC is to catalyze the computing research community and enable the pursuit of innovative, high-impact research.

In this episode of Catalyzing Computing, I sit down with Dr. Keith Marzullo, the Dean of the College of Information Studies, also known as the iSchool, at the University of Maryland College Park. He joined the iSchool from the White House Office of Science and Technology Policy, where he directed the Networking and Information Technology Research and Development Program, or NITRD for short. NITRD enables interagency coordination and cooperation among the over 20 member agencies, which together spend over four billion dollars a year in I.T. R&D. Keith is a member of the CCC Council. In this episode, we discuss his research, experience teaching, and time spent in the government. Enjoy.

Interview [00:01:14]

Khari: Keith, how are you today?

Keith: I'm great. How are you?

Khari: I'm good. So you've been on the CCC Council for a couple of years now.

Keith: I have.

Khari: How did you first get into computer science? Where did you grow up? How did you decide to study that?

Keith: Yeah. So I grew up in Southern California and I was interested in astrophysics. For me, programming was entertainment, something I could do. When I was in highschool, I taught myself Fortran IV and we had an IBM 1401 computer, which is like an ancient mainframe computer, generation two. And when I played with that I realized that I could have it run more quickly if I programmed it in assembly language -- it was called Autocoder.

That's why I got into the machine architectures of this extremely primitive machine, and so I continued to support myself through college programming. I did programming at my data processing facility at the college. I did programming at the Jet Propulsion Laboratories (JPL), things like that.

Khari: Okay, so what kind of programming was this like at the time? How does it compare to today?

Keith: Pretty simple compared to what we do today. I programmed up a meal card system so that students would be able to use meal cards to charge for their meals. That was a radical idea at the time.

[Laughter]

Khari: It's like a swipe card kind of thing?

Keith: Oh God, no. This was actually more of a fill in the bubble card.

[Laughter]

Keith: It was really primitive. We didn't have magnetic readers that were cheap at that time. It worked ok -- clearly technology had to catch up.

Then at JPL I ended up programming various scientific experiments. I did the spectrometer on the Viking Orbiter. They were looking for water, we were looking for water, and so I ended up programming the data analytics that went along on the spacecraft.

Khari: Oh, wow.

Ketih: And that was on a NOVA 18-bit machine. It was really great. And then I also did programming on the Voyager, did some of the data reduction on that. So I guess I'm involved in...my code’s not on the spacecraft, so it does not pass the heliopause, but at least the thing I worked on has passed the heliopause.

[Laughter]

Khari: Pretty cool.

Keith : Yeah.

Khari: So why did you switch, I guess, from like astrophysics to doing more of the computer science stuff?

Keith: Jobs. Getting a job in astrophysics was pretty hard at the time and I realized that I really love programming. I loved not just programming, but also thinking about computation. So I got pulled into some research at Stanford University, that's where I was a graduate student in astrophysics, and started working with a guy named Mike

Flynn. He was interested in different kinds of architectures, MIMD, SIMD at the time -- he came up with those terms -- and I enjoyed that. But then I got involved with Susan Owicki and formalism, how do you prove concurrent programs correct? And that was just beautiful stuff, so I got pulled into the formal side of things.

Khari: So while you're at Stanford, you worked with Xerox on the Xerox Research Internet?

Keith: I did, yeah.

Khari: Can you talk a little bit about how you got involved with that and anything interesting that came out of that?

Keith: Absolutely, that was a lot of fun. So Xerox at the time was inventing the Internet, their version of the Internet, the Xerox Research Internet, and I was part of the project called Pilot. Pilot was an operating system and I was a research intern, which meant that I had very little responsibility. As my boss said, they could fire me at any time.

[Laugher]

Keith: So it was quite liberating, like being a midshipman, you know. So I got to know the people doing communications. Alan Frier, Susie Armstrong, the whole group there, who were coming up with a suite of protocols that were in parallel to what we use now, like TCP/IP, but we were building this worldwide network. We had two Altos, these old kinds of computers, that were in the same room, but they were separated by 15 hops, which at the time was as large as you could make the network.

Khari: Okay what is a hop?

Keith: A hop was every time you went through a router, that was a hop, it would call a local area network. This is long before we had pairing agreements and things like that. And so we were able to experiment with, at the time, huge networks like, you know, thousands of computers.

Khari: Really?

Keith: Yeah.

Khari: Wow. And what year was this?

Keith: This was a... oh, my gosh... I got my Ph.D. in ‘84... and I was a Ph.D student for a long time, like nine years... So this would have been like 1979-1980.

Khari: That's pretty cool. So what kind of stuff could you do on this Internet? What was it used for?

Keith: Yeah, it was such a cool time. Research and systems goes through this virtuous cycle between having really great hardware and the software is behind, so you're trying to catch up, and then having really great software ideas and trying to have the hardware catch up. We were definitely in the first case. We were building these systems that were awesome in terms of hardware, I mean we had flat panel displays... or now they weren’t really flat panels, but they were bitmap designs, that's what I wanted to say. And we had servers, like file servers and print servers.

So it was distributed and we were netting things like, oh, remote procedure call -- Ruth Nelson was doing that work -- and so we were trying to figure out how to build these large distributed systems. Of course, that is now what we live in and the whole web architecture, we've well caught up to that. But at the time we were really trying to sort out what this new world would be like with a distributed environment.

Of course now...well we've rotated now and with the cloud, with 5G, with all these edge computing, edge devices, I kind of think we're in a similar state now where we have this incredible computing environment -- cluster computing, cloud computing, edge devices -- and we're trying to figure out what we can do with it. The answer is, of course, a lot. Everything from environmental monitoring to, well, everything in global commerce. And again, we're now at this point where we're sort of ahead in the hardware and trying to catch up in the software. This, by the way, came out of the CCC report [by] Mark Hill. Remember that report they did a few years ago on architecture?

Khari: Oh yeah.

Keith: It actually made that point.

Khari: 21st century architecture? I'm not sure if that's the exact name, but yeah.

Keith: That was a great report.

Khari: Yeah, so I was taking a look at your, I think this is your dissertation, about *Maintaining Time in a Distributed System*. So is this where you came up with Marzullo's Algorithm?

Keith: I didn't call it that.

[Laughter]

Keith: But yes, that's where Marzullo's algorithm came from.

Khari: Can you explain Marzullo's Algorithm and explain how you came up with your dissertation -- the process of doing that research?

Keith: Sure. So when I was looking for a dissertation topic, I went to one of my advisors, Hugh Lauer, and said I wanted to do distributed debugging. How do you debug distributed programs? And Hugh said, “Well, have you ever written a distributed program?” The answer was no.

[Laughter]

Keith: So he said, “Well, how can you learn how to debug it, if you haven't written one?” Which is a really good point. So he handed me a paper on clock synchronization and said, “Why don't you look at this?” And so I got pulled into the world of synchronizing clocks, which is a classic problem. At one point it seemed like everyone in distributed algorithms spent part of their life doing clock synchronization. So that's what I was doing.

I had the fortunate situation that I could build something that could be deployed on the research Internet. That was great. And after some discussions I had with the people at Xerox PARC -- Dave Bogs was really great in this -- I started thinking about synchronization as a kind of averaging; a fault tolerant averaging problem. And rather than thinking about clocks being point values, like saying it's 2:15, think of clocks as saying, well, it's between 2:13 and 2:17. So you have values, these errors, and the errors you can think of them statistically, and it's not a bad way to do so, but I was thinking about them absolutely. Saying, “If you take all the things you know about this system give me an absolute bound on what the time might be.” And of course, you might be wrong in those assumptions but you want to at least bound them in some way.

Khari: Right.

Keith: And so then what you're doing is you're looking at a problem of how do you synchronize intervals. How do you look at different intervals and find an interval that is the best estimate you can make given the constraints you have? On the assumptions you have, such as no more than two out of five values would be incorrect? That's what Marzullo’s algorithm is. It's an algorithm for taking intervals and a maximum number of failures and saying, “What is the tightest interval that you can compute based on that?”

It's like a version of voting. You know, if you think of the Von Neumann concept of voting, triple modular redundancy. If you have one possible erroneous line, then you triplicate it and two values that are the same are correct because you know, you have no more than one wrong. So I basically took that idea and generalized it to intervals.

Khari: So is this algorithm still being used for computing error in distributed systems like in clocks?

Keith: So when I talked to [David] Mills, who built up NTP -- this was a while ago -- it was still being used. Yeah, so the idea is still around, and it was embedded in some algorithms. Every now and then I get messages from people wanting to know more details about it and such. So yeah, I still think it's used. We've moved far past that. I mean, even some of the stuff we did, we generalize it to multiple dimensions. That was a lot of fun. And I think people are using more statistical approaches now. There's actually a nice way to take this work and have it work in a statistical way,

Khari: OK. So what would be the difference in terms of using it in a statistical way?

Keith: So this gets technical sort of fast.

[Laughter]

But if you can think of, say, a value that you read from a clock has some distribution associated with it. It lives within a certain value which can be long-tailed. And then you can look at the convolution of those values and ask what is the value given a certain probability. That's the statistical approach. You can actually also interpret this as fuzzy logic if you want to, that's another approach. They’re actually all related in some interesting ways.

Khari: So after you did your dissertation, what did you do next?

Keith: Yeah, I worked for a while at Xerox. Um, I ended up doing some work on software configuration management, we were doing all these software releases, and I got involved in that, but I was interested in going into academics. So I did that for maybe a year or two and then I moved to Cornell University as an Assistant Professor. I had made a friend, Fred Schneider, and he did a good job of selling me about the great things at Cornell University, despite it being in upstate New York.

[Laughter]

Keith: Actually, it is really nice and I think it is a great place to be. Not so much in the winter, but otherwise it's a great place to be and I really enjoyed my time there.

Khari: How long did you teach there for?

Keith: Oh, about six years, I think...yeah. And then due to issues of, like, jobs for my wife, we moved to San Diego and then I went to UC San Diego, where I was at for like, 19 years or something.

Khari: So what kind of classes were you teaching while you were at Cornell and at San Diego?

Keith: When I started, the chair at the time, David Gries, handed me a database class and said, “Keith, you're a systems person. Databases are systems aren't they? Teach this database class.” Which I did and I actually didn't know anything about databases, except I'd taken a class from Jeff Ullman at Stanford.

So that was fun, getting up to speed fast on that. In fact, there was this whole thing on normalization theory. I'm not going to get too technical, but this has to do with how you encode data and what happens when you do database operations, do you lose information or not. And I remember I just did not understand what Jeff was talking about. No idea, in fact, I don't think any of us did. So when I was teaching that at Cornell, I had to either sort of, you know, futz my way through it or really learn it, which is what I did. Actually, normalization is really beautiful, it's sort of like type theory in that...

Khari: What is type theory?

Keith: Type theory is how you, when you assign types to a language, how you're able to then induce properties of the program based on type correspondence. In this case, the types have to do with how you represent data. What is the format of the scheme of the table and what is the underlying semantics. So I got really excited and I told my students, you're going to love this. You're going to learn normalization theory and you're going to be amazed. And my students all hated it, just hated it.

[Laughter]

Keith: They couldn't understand a thing I was saying. So I sent a message to Jeff Ullman apologizing, saying, “I didn't realize it was so darn hard.” So then I ended up teaching operating systems, which, by the way, is more my area of interest. I taught that for several years.

Khari: So that's like how to build an operating system?

Keith: Yeah, it's certainly how to build an operating system. It's more how to think in terms of systems. Systems thinking is really important because you have to look at both the low level details and the high level details. They all come together, sort of the worm's eye view versus the bird's eye view of systems. Are you managing resources, or are you providing abstractions? And so systems is teaching those concepts, which operating systems are an embedment of, but there are other kinds of systems as well, indeed deep database systems are systems. David Gries was right.

My favorite part of it, though, was concurrency. Since I had done work in concurrency as a graduate student and understanding the beauty of the mathematics underlying concurrency was my favorite part of it. I don’t know if it was my students’ favorite part of it, but it was my favorite part.

[Laughter]

Khari: So after you got done teaching at San Diego, that's when you joined NSF or NITRD. Which one came first?

Keith: Uh, NSF did. Yeah, I had been chair for five years. We needed a new chair and it seemed like a great idea to get out of town and let the new chair have a complete run. I'd been talking with Jeannette Wing, she had been encouraging me to think about the National Science Foundation. So in 2010, I moved to the National Science Foundation as Division Director of Computer and Network Systems.

Khari: That's within CISE?

Keith: Right, that’s within the Computer Information Science Engineering Directorate.

Khari: So what kind of projects did you work on while you're at NSF?

Keith: So as a division director, it's sort of like being a department chair. So they'd like to say the best projects are at the program director level and that's sort of like the professor level. And it's true because they're the ones who get to look at all the grants, call up the person, say congratulations we're funding your grant, and all that great kind of stuff. And they have a lot to say in the science. The next level up, we do a lot of program work. How do you come up with new programs? For example, I don't know if you've ever read the federal budget? It's riveting reading.

[Laughter]

Khari: Certainly, not in its entirety.

Keith: Yeah, I don't think anyone does. But there is part of the budget that's the National Science Foundation and in there you have, like, what is NSF going to do in three years and five years and so on. I wrote the text for CNS. That's what division directors do. We work with the directorate to plot out the directions of the division and therefore the direction of CISE.

My favorite project was one of the earliest ones. The White House through OSTP, the Office of Science Technology Policy, was encouraging us to develop a joint program in cybersecurity with Social, Behavior, and Economic Sciences (SBE), that directorate, as well as with Mathematics and with the Office of Cyber Infrastructure. The idea was -- due to us changing the plan they published -- was that there were economic incentives that one could use to help change the basis of cybersecurity and that's true.

Everything from insurance, to understanding economic principles underlying the kinds of things people do, and so I was given the task of working with my colleagues over in SBE to craft a program that would allow us to share funds or at least share the reviews so that we could fund things together or fund things in a coordinated way. And the two cultures are really, really different. Social scientists, computer scientists are quite different and trying to come up with a program that would work across both divisions was a challenge.

Keith: What kind of differences do you see across those areas?

Keith: At a very high level, and I'm going to be oversimplifying here, but just to make a point, computer scientists like to solve problems. They're for the most part engineers, some people are mathematicians, but for the most part engineers. They want to come up with a cool solution to a problem. And social scientists are scientists. They want to understand things, they want to have theories and make hypotheses and see what they can conclude from that. And solving problems and understanding science, while they can work hand in hand, they really have a different set of values.

I think the social scientists often look at computer scientists and say, “Where's the theory? What are you learning from this?” And the computer scientists may look at the social scientists and say, “Well, who cares? Don't you want to solve problems? Don't you want the world to be a better place?” Things like that. That in a nutshell is the tension between the two.

Khari: So then in terms of the proposals you were receiving, were there big discrepancies as far as what people wanted to address or or how they wanted to go about tackling these problems?

Keith: So...yeah, and there were certainly proposals that went into the social sciences, and they funded those, and there were ones that went into CISE, and we funded those, but the trick was trying to find proposals that we could fund together. We created this one program that I think worked very well towards that.

NSF has a vehicle called EAGERs and I don't remember what EAGER stands for... Early Research… or I don’t know. Basically, it's a proposal that you can write. It's for no more than two years and no more than $300,000 and it's decided only on the basis of a program officer -- it doesn't have to go to a panel. They're meant to be a vehicle for funding, higher-risk, higher-reward research that wouldn't do well in a panel because it's too off the wall, but if you did it right, it'd be great. So we created an EAGER program in which we engineered it. We said you had to have some social scientists and you had to have some computer scientists, and they had to have not worked together before, and they had to propose a joint project. Out of this then we got some interesting conversations going on between social scientists and computer scientists.

We would first take white papers, and out of those we would accept about half of them and those half would then go on to submit proposals and out of those we would take about half of those. So roughly 25% of each attempt would get funded, which is not bad, and there was feedback the whole way so people could try again. And the results were good. We got a lot of good papers out of it, new partnerships out of it, and they were doing really interesting things.

One project was to put ethnographers -- so these are social scientists who study how people behave -- and they put them in security rooms, basically groups of people who are trying to keep the systems of business safe and secure and understanding what were the rules they were using and try to write that down because much of the activity they were doing is not written down, it's just behaviors. And it was kind of cool. There was also things on passwords and such as you'd imagine.

Khari: So what was the process, because you said people had to not know each other first. What was the process to pair people up?

Keith: Oh yeah, if I said that, no, that isn't quite right. They couldn't have published together.

Khari: Ok.

Keith: So obviously they knew each other. Yeah, that would've been cool… take people at random. No, but they had to know each other, just that they couldn't have published together. The idea was to create new collaborations.

Khari: So then from NSF you went to NITRD?

Keith: I did, yeah.

Khari: What was the decision behind that move?

Keith: When I was at the National Science Foundation, at some point I converted over to being a Fed. So normally when you go to NSF -- most people when they go I should say, not normally -- they go under what's called an IPA, interagency personnel agreement I think is what it stands for, and what it basically does is it allows you to stay working at your old job like a professor and then they pay your salary by going back and effectively giving what looks a lot like a grant to your university. So your salary is covered, your health care is covered and everything, but you're now full-time at NSF or some other agency. Other agencies do this too. But at some point I converted over to being full-time. You can only be an IPA for up to four years and I was really enjoying what I was doing, so I converted over to being a Fed.

Shortly after that, there was a need to have someone take over NITRD. George Strawn, who had been running it for a while, was about a year from retirement and so they were looking for ways to get someone else to take it on, and since I had done so much work in the interagency anyway -- through my job as Division Director I had done a lot through our Cyber-Physical Systems program with other agencies -- they thought I might enjoy it, thought I might be good at it. So I applied and I interviewed and I got the job.

So it's a very different kind of job, right? Working at NSF you're really close to the science, at NITRD you're really close to agencies. You're working at the interagency and you're trying to find ways to have different agencies align what they do, be able to cooperate, to be able to do things together, and agencies have really different cultures. NSF people are very different from NASA people, who are very different from transportation people, who are way different from education people. They all have their own rhythms, their own way they think about what they do and so in NITRD the idea is to get these groups together to cooperate on things like big data, networking, cybersecurity, privacy, robotics, things like that.

Khari: Ok, so what would the day of the NITRD Director look like?

Keith: Like everything in the government, hectic. NITRD runs a lot of these working groups and so some of it would be me going to working group meetings just to understand what the agencies were doing -- remarkably little of that.

Some of it was talking with people in different agencies to see what we could do to start new activities. Part of it was over in the White House meeting with other members of the office of technology policy, there's other coordination groups to see what we could do together. Gosh, it's a blur.

[Laughter]

Keith: We were moving really fast at the time. This was near the end of the second term of President Obama and he was running through the tape, trying to get everything done before the administration was handed it over to the next president. And so part of what we were doing was getting these strategic plans written in cyber security, in privacy, in big data and artificial intelligence and I was doing a lot of the, kind of, getting the people in the right places to do those activities successfully.

Khari: Ok. How do you think computer scientists should interact with policymakers, who might not be experts, but need to know to make critical policy decisions that will impact how technology interacts with society?

Keith: So how is an interesting question, you can interpret that in different ways. I mean, procedurally, how do you get the knowledge of a computer scientist in front of policymakers, that can be hard. Latanya Sweeney, a professor at Harvard, has some great ideas on that. Much of them involve training computer scientists to go work in places like the FTC, who are basically a bunch of lawyers, or at least they have a lot of lawyers on staff, and having the technologists be able to inform those lawyers about technology.

So the example that Latanya uses is that if I record you with my smartphone it's ok for me to visually record you but to record your voice that depends on the state. Do I need your consent or not? And so the laws as they are currently written don't mesh necessarily very well with the technology, so having someone who understands what can or cannot be done is very important to policymakers.

Khari: I did not know that. So you can record someone visually anywhere -- I mean I assume outside of the bathroom or whatever -- in a public space, but not necessarily the audio?

Keith: Audio recording is more ticklish and more sensitive than video recording, if I remember the argument Latanya made. Wwe're gonna look that up…

Yeah, that's true. The legal structure for recording someone depends on the state and I think that the way Latanya explained it to me is the audio part.

Khari: [From] Homevid.com [note: website no longer exists], I don't know what that is, but [it says that] in general, most video-only recordings are legal whether

you inform the persons you are recording them or not provided their privacy is not invaded.

Keith: Just what you said.

Khari: Recording audio without the subject’s permission is almost always illegal, but may be legal in certain situations.

Keith: There you go.

Khari: Yeah, that is actually pretty crazy, I guess. Yeah, that does not reflect the current state of technology at all. So that definitely seems like something that maybe should be updated or at least considered how that has evolved.

Keith: Yeah, there was this other... Well there are a lot of examples of where policy decisions are made that don't reflect technology and we clearly need to find ways to increase those conversations.

Outro [00:28:03]

Khari: That's it for this episode of the podcast, I hope you enjoyed it. Tune in next week as I continue my conversation with Keith Marzullo as we discuss his current work at University of Maryland's iSchool. We also discuss the impact of technology on society and the role the social sciences has to play in designing computing systems. Until next time. Peace.