Oral History of Brian McCune

Interviewed by:
Hansen Hsu

Recorded May 16, 2018
Mountain View, CA

CHM Reference number: X8636.2018

Table of Contents

Hansen Hsu: Today is May 16, 2018. My name is Hansen Hsu, curator at the Computer History Museum. I'm conducting an oral history interview of Brian McCune. This is part of the ongoing oral history program of the Software Industry Special Interest Group, which is part of the Computer History Museum in Mountain View, California. We are here at the Computer History Museum. This interview is being recorded and videotaped and will be transcribed, edited, and posted on the Computer History Museum website. Let's get started.

Brian McCune: Okay.

Family and Early Life

Hsu: First things first. Where and when were you born, and where did you grow up?

McCune: I was born in Portland, Oregon, and I grew up there for until I went to college, for 18 years.

Hsu: What did your parents do? What was your background?

McCune: Well, my mother was a homemaker, and my father started out as a lumberjack, because in the early days of the 20th century, there was still a lot of timber in Oregon. Then he moved from that to being a roofing and siding and home remodeling contractor. In fact, he ended up being the second-largest roofing and siding contractor in Oregon in the 1940s, 1950s, 1960s. So pretty low tech.

Hsu: What were your interests as a child?

McCune: I had a great childhood, and I had an early helicopter soccer mom for a mom. She got me into everything you can imagine, from swimming to Cub Scouts and the Boy Scouts and programs at my church. Probably my biggest interest has always been music, so I was in the boys' choir at my church from age eight. Then later I was the only kid in high school allowed to be in the main church choir, and then I was in choirs in high school. Oddly enough, I played Lancelot in "Camelot" in 1967. That's probably one of the peaks of my career, or that side of my career.

Hsu: Was your family particularly religious?

McCune: Well, yes and no. I mean, we went to church every week. My father was a trustee of our church, so yes. I'd say we were fairly religious but nothing special, especially for the 1950s. I mean, pretty much in the United States everyone went to some church, I would say.

Hsu: Right. Any particular denomination?

McCune: Yes. We were members of Westminster Presbyterian Church in Northeast Portland.

Hsu: What were your favorite subjects in school?

McCune: Pretty much everything except art, handwriting, and PE. My brain is ambidextrous. I'm very good at math and science, but I'm also very good at languages and writing and comprehension and, as I mentioned, music. I really loved everything. I was probably best at math and science, but I wouldn't say that I liked them to the exclusion of everything else. That came later. That came in college really.

<u>First Exposure to Computers</u>

Hsu: Did you have any influential teachers?

McCune: Well, as far as how I got into the computing industry and into AI and expert systems research and then development, it all started in my sophomore English class with Mrs. Allen. Unfortunately, I can't remember her first name. But that year in high school, you had to do an essay on "What Do I Want to Be When I Grow Up?" because they're trying to get you to realize that, as a 15-year-old or 16-year-old, the gravy train ends soon. You're either going to go to college or trade school or getting a job, whatever. I don't know. I honestly don't know where, but I said, "I want to be a computer programmer."

This was in 1965. Now, I will tell you, there were fewer than 12 computers in the entire state of Oregon at that time, so I don't know. I must've been reading about computers in *Scientific American* and *Popular Science* magazines and those sorts of things.

I found out that there was a girl in my class and her father ran the computer center for the largest bank in Oregon, so I got him to give me a tour. I found a computer. That was step one.

Hsu: You mentioned you had been reading *Popular Science* type articles. Was that how you discovered computers? Were you into science fiction?

McCune: Somehow, I knew that they existed, but I didn't really know much beyond that, but I'm a very logical person. That's why I got into artificial intelligence. I like to organize data. I like to figure out how you can search data of all sorts—signals, images, text, relational data, any sort of data. That's always been true. I used to read the encyclopedia for fun when I was in the second or third grade. You know, I was just strange, but somehow, I knew there were computers and I knew that if I could find some and learn about them, I would be interested in them. But exactly how it happened prior to this essay I wrote, I can't remember. There was no one giving a lecture on the future of computers in high school, so I'm speculating on the fact that

I just must've read it because I read lots of magazines and newspapers and things like that. Somehow, I was aware, but very shallowly aware, as you'll see.

Later that year, Robert Byers, who was my chemistry teacher, said, "You seem to have aptitude for science. Follow me." One day after school he took me to Reed College, which is a very good liberal arts college nearby, and he said, "Here's a computer." At this point, this was the second one I'd seen. This was a little box with a wavy line. Looked like an oscilloscope. Well, it was an analog computer with a patch panel. You could do integration and differential equations and things like that and see the answer come up on the scope.

Now, that didn't interest me all that much. I really wasn't interested in physics and continuous equations and things like that. I was very much more interested in discrete things like first-order logic, which is the equivalent of an SQL database query. You know, the difference between the Dewey Decimal System and the Library of Commerce System, Congress catalog system. Those sorts of things interested me. How do you organize data? When I saw this, we played around with it for few hours. I said, "Okay. I'm done with that. I got to find me a digital computer."

Hsu: It was very clear to you the difference between analog computing and digital computing just from that experience?

McCune: Oh, yes. Oh, yes. Now, again, I didn't really know what it meant to program a computer. I was sort of guessing at that point, but the next thing that happened was in my junior year, the very next year. My high school, Grant High School, had a very good reputation in the 1950s. In 1954 it was named the Best High School in the United States by somebody. I don't know if it was *Life Magazine*. Who knows? It's one of these popularity contests that are somewhat meaningless, but it was now on the radar screen of the University of Chicago, one of the greatest universities in this country.

They had a National Science Foundation summer program for juniors, so we got to send four people. My best friend Marty Schnitzer and I got to go back there for two or three months. We each took three classes, and one of my classes was programming an IBM 7040 in Fortran. That's where I learned how to program, and the course was taught by the president of the ACM, who was a professor.

Hsu: Wow.

McCune: I think he was actually the president at that time or maybe just a couple years later. Very famous guy, Robert Ashenhurst, and he was actually a professor in the Business School.

This is was in 1966 or 1967, and there were no computer science departments. Stanford started theirs in 1965. Most computer science was done out of mathematics or electrical engineering. In this case, Ashenhurst was in the Business School at Chicago. I learned how to program, and then after that, I was hooked.

The next thing, the next year, my brother came home. He was four years older than me. He had been a student at Williams College back east in Massachusetts. He came home and he said, “I’m transferring to Oregon State University, but this summer I’m going to do some programming.” One night he dragged me down to Portland State College—it’s now Portland State University—in downtown Portland. He said, “Here’s a 1620, and you can have a hands-on experience.”

At the University of Chicago it was key punch a bunch of cards and submit the deck and come back the next morning. If you think about debugging cycles today that take place in a matter of seconds or minutes to find a bug and change one bug, we got to change one or a small cluster of bugs once every 24 hours. You better design your program and not make a lot of obvious mistakes. It really changes the paradigm because to get anything done you’ve got to quickly converge to a program that works. That was a classic punch card operation.

But when I got to the 1620, this was you have the whole computer. There were no computer screens, and there was still a card reader, but you could put anything you wanted in the card reader, wait until something came back, put it back through. You could suddenly get a cycle going of debugging every, you know, maybe 10 or 15 minutes instead of once every 24 hours. It’s still not interactive timesharing as with your PC, but it’s one small step in that direction. That was a lot of fun.

Hsu: Your brother’s name is Mike?

McCune: Yes, Michael McCune.

Hsu: He was also interested in computers.

McCune: Oh, yes.

Hsu: Did he get into it before you did? Is he older, younger than you?

McCune: He’s four years older, and I assume he had learned about them prior to this, because he would’ve been about a junior in college at that time. As far as I know, this course he took at Portland State was his first programming course, but he really got hooked. By the way, he started companies. He started multiple companies just like I did, and he ended up being the largest website developer in Northern Thailand, both physically and in terms of a client base. That was his last company.

Hsu: Hm, interesting.

McCune: Yes, he and I are very much alike. You know, very logical, perfect computer science candidates.

Oregon State University

Hsu: Then you went on to college at Oregon State?

McCune: Right.

Hsu: Any reason why you wanted to stay in state?

McCune: I think my parents wanted me to stay in state, because I got accepted to Stanford. Early acceptance. I got early acceptance at Pomona. These are two big-name schools. There had been a whole bunch of riots at Stanford and at Berkeley right before that.

Hsu: Oh, this was during the Vietnam era?

McCune: Right. Vietnam protests. Well, turns out we had them in Oregon State too, but they didn't know that. They came later. I think they simply didn't want to lose me into the cesspool of California politics and hippiedom because this was 1968. Fall of 1968, and things were a little crazy in those days. We also had the race riots in all the big cities around the country.

We also really couldn't afford to send me away. We didn't have the federal loan programs in those days that would make it easy to go pretty much anywhere. I didn't get a job or a scholarship, so I was going to have to really work hard and save money and then borrow money from somebody. It was just easier to go to Oregon State, and I planned ahead. I said, “Okay. I'm going to go to Oregon State and then I'm going to Stanford for graduate school.” Now, when you're 18 to say that, most people, including me, would be like, “Give me a break. You're not going to get into Stanford.” I just said, “Well, that's my plan,” and in fact, it did happen, so it worked out well.

Military Deferments in the late 1960s

Hsu: Did the politics of the time, the Cold War, the Vietnam War, affect you or your family any at all?

McCune: It did not affect me. It actually affected my brother Mike because when he got out of Oregon State with his master's in mathematics he was going to be drafted immediately, because he had had a deferment. By 1968 or 1969, as soon as you left college, they figured out when you turned 19, whether they wanted to draft you or not, and then they deferred you. But as soon as the deferment was up, you were drafted.

McCune: The only way he could avoid going to Vietnam was to find another deferment. He moved to Los Angeles, tried out for a singing group, Up With People, and for whatever reason he didn't go with them. They were sponsored in part by the Department of State of the federal government, so they would have gotten him a deferment. They would go all around the world for a year. That was the folk song era, so this was sort of the feel-good side of popular music as opposed to hard rock that was going on in the late 1960s.

He didn't do that. He found a job and went to work for Litton Data Systems, which is a defense contractor, and immediately, bingo, he got a job deferment.

Hsu: Right. Okay. If you work for defense, something that helps the war effort, you can get a deferment.

McCune: Yes, that's right. But none of that, affected me, because I was deferred. I was in college.

Computer Courses and Exposure at Oregon State

Hsu: Right. Okay. So, yes. Let's talk about your work at Oregon State.

McCune: Okay. Well, I majored in mathematics because, as I said, nobody had a Computer Science department, although Stanford had one. There were probably four or five in the country by then. My advisor was a guy named Harry Goheen, who was a really great teacher and a great advisor, great guy. Slightly wacky, but very good for me. He was mostly into the theory of computation, and his background is he came from the University of Pennsylvania, where he worked on ENIAC, the first computer. Let's not go down that rat hole. Anyway, they claim it was the first computer. But while he was there, 1947, he was one of the founders of the ACM, so he had this rich tradition, even out at Oregon State, of innovation in computing. He was my advisor.

Then my other advisor was a guy named Arvid Lonseth, and he was actually in applied mathematics. I took classes from him, got to know him, and we became friends. But the interesting thing was he had been a graduate student with George Forsyth at UCLA [University of California, Los Angeles]. Now, George Forsyth was the founder of the Computer Science Department at Stanford University, and that's actually why I ended up going to Stanford, or at least the initial reason, because I felt I had a connection there.

This is when I started taking only computer classes. Even though they were in the Math Department, there were a few dozen classes, and I took them all. In fact, after the first class that I took the first quarter, which is programming in an interactive programming language called OSCAR of all things, they said, "Gee, this guy seems to know what he's doing." They asked me to become a lab assistant as a freshman in college. So, I did that, and then after that the computer center said, "Well, we want to hire you as a programmer." You know, why be a teacher when you can do real work?

I thought, "This is great idea," so I said, "Yes." Then they said, "Oh, we're so sorry. We can't give you that job yet, but we'll give you this temporary job," because they knew that they would lose me, I think. I don't know. So, I became the first male IO girl. Now, this is the age of sexism, so there was this one job category called IO girl—that's what we called them.

Basically, they stood behind a counter, and as you brought your decks of cards up, they took them in and handed them to the qualified computer operator, which I was not. They ran them through the card reader, ripped off the line printer output, and came back out. Then you would tie the output around the box, put it in an A through Z cubbyhole, and wait for the person to come back and pick them up. That was the job of the IO girls. I like to say I broke the sex barrier in computer science in a strange way.

Hsu: In a strange way. Yes. Well, I mean, I guess the real question is whether any of those IO girls advanced to other positions in the computer center.

McCune: I think so. Some of them not, but some of them were in the same boat I was in. They were trying to become programmers. They were students working part time, whatever. It was clearly a position you wanted to advance out of. Quickly.

Hsu: Right.

<u>Programming at the Oregon State Computer Center</u>

McCune: I only did that for, I don't know, maybe a couple months, and then I hit maybe the best job I ever had, and I discovered that I'm not really a long-term researcher where I pick a neat area and I work my whole career in that one area. I'm a problem solver. I guess I would say that I'm a consulting engineer. I like people to bring me problems. I solve their problem, then I move on to the next one. That keeps me refreshed. The problems might have some relationship to past problems that I've worked on—just a very stimulating sort of environment. So, in a way, starting with this next job, my first programming job, that's what I've been doing for 40 years, 50 years.

Hsu: What sort of job was this? You were still an undergrad at the university?

McCune: Yes.

Hsu: This was with a lab, with a center?

McCune: This is for the central computer center. Again, even in those days, about 1969, we only had maybe a dozen computers on campus. I programmed for the big mainframe, which was a Control Data Corporation 3300.

Hsu: That was a timesharing system?

McCune: As it came out of the box, it was not. It was a traditional mainframe with card readers. But we had an enterprising group of guys under a staff member named George Rose, and they developed, from whole cloth, one of the best timesharing systems in the world called the OS3 or OS^3 really. Open state, open shop operating system. We had a couple hundred teletypes scattered around campus, and people didn't need to type in on punch cards any longer. You could type your program in from the teletype, run the program on the mainframe, and get the answer to that.

I started working for a guy named Ron Davis, and basically, whenever someone came in with a difficult job, he said, "Go meet with Brian." Then I would program it up or often they'd have a program that needed to be modified or maybe even just, "Here's this year's data, run the data, and if it looks like it worked, give it back to me." I did dozens and dozens of jobs. I was a full-time student. I worked full time in the summers and during the main year half-time, three-quarters time.

Oregon State Sensor Data Applications

McCune: I was pretty busy, and during that time, I got interested in what we talked about earlier, which is data reduction, data analysis, and particularly I worked on sensor data. I had an application. I listed some of them here. I had a system for the oceanography department, where they had sensors every quarter mile starting at the high tide mark of the Pacific Ocean down in Newport, the beach, Oregon, and they went every quarter mile out into the ocean. They were looking at what lives in the inner tidal zone and what sort of temperatures and what sort of tide flows there are. They gathered this data, and then I had to analyze it basically. I mean, they gave me the theoretic algorithms, but I had to write the program and actually run it. That's an example.

I had another system that had sensors. Oregon State has one of the best forestry schools in the world, because it has a lot of forests, and they put a sensor every 10 feet up a 200-foot tree, again, to see what the wind is like, what the temperatures are like, what animals are living at that level in the tree, and so and on. This is in 1969. I mean, this was not done. This was pretty far-out stuff, so I processed that data.

I had another database for the DEQ, Department of Environmental Quality of the state of Oregon. I had a data record on every mile of every river and stream in the entire state, and they would go around with sensors and figure out. They'd also see what cities, homes, and factories were there that were polluting because we had a horrible polluting problem. In the early 1960s, the Willamette River caught on fire.

Oregon became green overnight. We decided, "That's a bad thing. How do we fix it?" This database was one of the ways we fixed it. We tracked every polluter, and we could tell if there was a paper mill pumping out polluted water from their papermaking process. I could tell you every mile from that mill down 200 miles how much pollution was still in the water as it got sifted out. That allowed the state to say, "We got to close that company down," "have more pollution controls," or what have you.

I had another database with a record on every quarter mile of state highway. That was more for maintenance purposes. They would go around and say, "This concrete's breaking down. Within the next five years, we've got to repave." That sort of thing. This was sort of the era when computers were starting to be put into use in every walk of life, not just big banks and, you know, nuclear labs and things like that.

Hsu: What sorts of languages were you writing these programs in?

McCune: Mostly in Fortran, Fortran IV. That was the go-to language. I mean, I knew many other languages: Algol, Cobol, Basic. This homegrown interactive language that I think was based in part on Basic was called Oscar, the Oregon State Conversational Aid to Research.

I think pretty much everything I did at Oregon State was in Basic or it was in Fortran. Yes. It was the best scientific language. It was efficient and everything.

<u>First Exposure to AI</u>

Hsu: Is there anything else you want to talk about during your undergrad?

McCune: Oh, yes. How did I get into AI? Very important. Oregon State didn't do artificial intelligence. I mean, really, nobody did. Even though I said there were 20 classes I took, there were no AI classes.

This rather famous mathematician, Emilio Gagliardo, came over from Italy and spent five years at Oregon State, and he gave a graduate seminar in pattern recognition and I took that. Then I was hooked not only on computers and programming but on artificial intelligence. That was pretty basic artificial intelligence, but still. Compared to what I had been doing, it was like night and day. I mean, dump in data and find hyperplanes. Basically—what's it called? —it was bottom-up learning. It's basically cluster creation.

Hsu: I see. Pattern recognition was already considered a subfield of AI at that time?

McCune: Well, okay. Now you get into the philosophy. There are lots of different decision-making technologies, and they often have started up in different departments. The EE [electrical engineering] people would say, "That's just an EE technique," but AI certainly embraced it because it was a way to do machine learning—a fairly primitive one, but perhaps the first one, but yes. I wouldn't say it's definitely artificial intelligence or not, because other people would say it's not.

Hsu: Right. What got you excited about that course? What about it got you excited? What you were doing as AI at the time?

McCune: Ah. That's an important question. Yes, I think so. I think the course was not called AI, but I think it came up that there's this field called artificial intelligence. This is the only thing we do in it here at Oregon State. So yes.

Hsu: Right. That course made you aware of the field of AI.

McCune: Yes, absolutely. In fact, that was my junior year, and that summer I bought my first car, a lime green 1971 Camaro. My best friend and I drove to Tijuana, mostly just to test out the car. But along the way we stopped at Stanford because a guy who had been at Oregon State with my older brother Mike, Lynn Quam, now worked at the Stanford Artificial Intelligence Laboratory.

I said, "Next year I have to apply for graduate school." So, I went in and met with him, and he gave me a tour of the AI lab. Then I was really hooked because there were robots that moved and arms and cameras that produced digital data that got analyzed and all sorts of weird. I mean, there were automated vending machines, so you could go up to the vending machine, type on a computer your name and password, and it would open the thing up. You'd take a Coke out, and it would send you a bill at the end of the month.

We had a lot of toys. Really cute toys, and I'll come back to one of them later, because it ended up being key to what I did later. These things are very incremental. Learning about computers, programming pattern recognition, AI. Every year I learned something new. There was nobody that said, "You need to go in artificial intelligence." In fact, some of my advisors saying, "You don't need to go to Stanford. Go to a good database school or someplace." You know, Ohio State or I can't remember where they wanted me to go, and I thought, "But they don't do AI."

Stanford University

Hsu: Okay. Then you apply to Stanford, and you had this connection with Professor Lonseth already.

McCune: Yes. But he just wrote a letter of recommendation. I got accepted. Went down to Stanford in 1972, and Stanford just sort of took everything in Oregon State to another level. Not only did it have AI, but it had the most funding in all the areas that Stanford had going, and it had the best person. Other than maybe somebody at MIT or Carnegie Mellon, it had the best person in every field in the country.

It was like a kid in a candy shop taking classes for the first couple of years. I learned a lot from Don Knuth, Bob Floyd, and Ed McCluskey in hardware design and computer architectures, which I didn't go into, but it was like another view of the computing substrate, if you will. I really enjoyed him. Plus, they're all characters. Goes with the territory if you're in computing and you're that smart. You're always a little quirky, present company excepted, of course.

I loved taking classes from these guys. You had an academic advisor. My first year was Harold Stone, because I had declared that I was in software systems, not in AI yet. I went there with a National Science Foundation fellowship, so I didn't need any funding. I was self-funded.

In fact, I'm not sure if it was that reason or just my interest in AI, but I got a call from Ed Feigenbaum that summer saying, "Come to work for me," which I didn't do. Maybe I should have, but by then I'd already been at the AI lab, and I said, "Do you have any robots that move around?" "No, no, we don't do robots. We do expert systems." I said, "Oh, that's not as exciting."

It's funny, you know, when you're 18—no, I guess I was 21 then—think how you make decisions that affect your life. Although as you'll see, Ed had a big effect on me. In fact, that's really why I'm here today, because of Ed Feigenbaum.

So, my first advisor was Harold Stone. He was in software systems and computer architectures, and after a year, he left Stanford and they randomly assigned me to an interesting guy named Vint Cerf. I wasn't in networking at all, so I interacted with him and knew him really well. He's still a friend of mine. But I was not going to do a networking thesis again.

Maybe an opportunity missed, I don't know, but I started working with a guy that year named Cordell Green. He was a hardcore AI guy. He had been a student of John McCarthy's in the 1960s. Got his PhD at the AI lab and was also working at SRI International. He won the Grace Murray Hopper Award as the best computer scientist under age 30 for something called QA3, Question Answering 3. He showed how you could use first-order predicate logic to not only prove theorems but simulate programs, automatically create programs, prove the programs are correct. It's sort of like he took John McCarthy's ideas and created a whole field, which actually then led to my PhD thesis, among other things.

Natural-Language Programming Research

McCune: He was very seminal. One of those areas was called program verification. That became a whole subarea of mathematical computation, theory of computation in AI. But another area was automatic programming, which then became an area called program synthesis, which is around today, and Cordell's group is still in Palo Alto. It's called the Kestrel Institute. We had DARPA [Defense Advanced Research Projects Agency] funding, and we had a very large project called the SY system that went from 1974 to 1979 roughly, and seven PhD theses came out of that. Mine was one of seven.

The idea was you should not need to have a programming language to program a computer. You should be able to type in English sentences. It should ask you a few questions, and eventually it writes the program for you. It does all the dirty work. Well, turns out, even today, that's easier said than done. We spent years doing that, and it was successful but a very big and ponderous system, and the work continued for decades. It's a very small niche, but an important niche in artificial intelligence and in programming languages and tools as well.

Hsu: You mentioned that it was successful. The ideas were sound but maybe the implementations were just too slow?

McCune: Yes. Well, a couple of things. One is that the vision was so vast. I mean, going from natural-language statements, English statements about a program, to super optimized code. That's why there were like seven PhD theses along the way. We couldn't even run this program on the PDP-10 at t the AI lab, so we ran it on PDP-10s down at the Information Science Institute of USC. It was all built in; well, it wasn't common Lisp. I guess Interlisp, which is a system that came out of BBN and then transferred to Xerox PARC, when PARC was started around 1970. Each one of these programs was like a million words of memory, and there are like seven of them interacting. We only ran it a couple of times as a complete end-to-end system, and it was slow. We took over an entire multimillion-dollar computer and it ran for hours, and it worked. From a research point of view, we think we proved the case, but it was ponderous and so on and so forth.

In fact, at Systems Control, the next company, we started a new follow-on program called CHI. A guy named Jorge Phillips, another graduate student, started that, and that created the interactive refinement technique, where you use predicate statements and then the system automatically refines those into further and further detailed refinements of predicate language, predicate logic. Eventually it starts turning into procedural code. That worked. I mean, that was efficient. The next generation really was in some sense more practical.

Hsu: Ultimately, why do you think that idea still hasn't caught on?

McCune: Well, I think the main problem is related to what we were talking about the last two days in this expert systems workshop that the Computer History Museum has sponsored. It's a big, big program. I tell people computer programming is perhaps the richest intellectual exercise and the least forgiving intellectual exercise there is. If you're doing artificial intelligence research, it's not where you start.

It's sort of like you don't start by saying, "I want to build a computer system that writes the top novel of the year and wins the Pulitzer Prize." That's not where you start. That's stuff that most people can't do. It was just a very, very hard problem, and it still is.

Dissertation Research

Hsu: Was your dissertation on this topic?

McCune: Yes. It was on something called a program model builder, which sat in the middle of this big system. You input natural language, full sentences. Those got parsed into smaller chunks. My model builder assembled those into an executable program in what is called a very high-level language, a set-oriented language, and then from there we had systems that refined the high-level statements—this is the early version of the refinement process--down into executable.

Actually, it was Lisp in those days, and there was another model that was basically Don Knuth in a bottle. It tried to look at each piece of the program as it was being refined and tried to assess which implementation technique. If what you needed was a linked list, it might have five ways to implement linked lists. It needs to know how often you look something up, how often you add something, because the optimal code would've been on which of those operations is most frequent. That's why I say that's sort of the stuff that Don Knuth analyzes rigorously in the analysis algorithms work that he pioneered.

Hsu: Okay. You graduate in 1978?

McCune: Well, I graduated the end of 1979.

News Service Application

McCune: Let me mention one thing that happened along the way. As I mentioned, John McCarthy liked toys, because he felt if you had smart computers and smart people and lots of sensors and lots of robots, some bright graduate student's going to figure out what to do with them. He got the Associated Press to give us the first real-time link of AP articles. Every time an AP article came into a newsroom, the article would get typed out on a teletype. You'd rip it off, take it over, and they'd retype it and put it in their typesetting system, if the newspaper wanted to run that article.

Well, we just dumped them all into a big text database. Then John worked with a guy named Martin Frost who wrote an application to do keyword queries against this database. That was called the News Service Application, or NS for short.

I became a power user of NS. I knew what was good about it and what was wrong about it. It will come up later why that turned out to be important. It was John's idea of let kids play and good things will happen. And this did happen, although it happened after I'd graduated.

Hsu: Okay. That was something that you were sort of doing on your spare time while in the AI lab?

McCune: Yes, yes. I mean, I basically, I read the news; I didn't read a newspaper. When I got to the AI lab, which unfortunately was like at 10:30 or 11:00 in the morning because we worked all night to get more cycles on the computer, first thing I'd do was read my e-mail and then I would read the News Service. I had some standing queries about things like artificial intelligence and I don't know what else I was interested in—maybe sports teams. I developed this set of queries, and every morning I would see what was there, and that essentially gave me a custom newspaper to read. Now, this was in 1973, so long time ago.

I mean, nobody had the data. It's sort of like we talked about this yesterday. A lot of these algorithms existed in the 1950s and 1960s. There was either no data or if there was enough data to make the algorithm converge to a reasonable answer, the computers were not fast enough to process that much data. We had the data. We had the Associated Press newswire, which in those days was fairly robust. It had maybe a few hundred articles per day, covering all sorts of topics.

Systems Control Inc.

Hsu: Next you got to Systems Control, Inc.?

McCune: Yes. What happened is my thesis advisor, Cordell Green, decided to leave Stanford. Our project was part of John McCarthy's overall grant from DARPA, the Defense Advanced Research Projects Agency, although in those days it was probably only called ARPA. Cordell went to DARPA and they said, "Yes, we'll continue to fund you. Let's find you a home," and we found a home in Systems Control, Inc.

The reason we found a home there is that Ed Feigenbaum and his wife Penny Nii had two years earlier—or no, just a little bit earlier… I'm not sure what date, but before 1978, they had taken a project where they applied the Hearsay-II blackboard system architecture in artificial intelligence that came out of the Hearsay Speech Understanding System at Carnegie Mellon University, and they were applying it to SOSUS data. SOSUS data are these undersea microphones that the U.S. scattered around the oceans that tried to pick up noises from Russian submarines during the Cold War. The idea was you could use the Hearsay-II architecture. Instead of listening to someone talk and typing out a sentence of what they said, you would listen to all these sensors and put up on a graphics screen a map of what's moving and label it—submarine, whale, unknown, U.S. submarine, Russian, you know, whatever—however well you could identify that object from the array of acoustic sensors. That was being done at Systems Control, Inc., or SCI, and I think that's one of the reasons that was funded by DARPA.

DARPA said, "Well, why don't you move there?" and we did. I know one of the things I did while I was there was we wrote a proposal to the National Science Foundation, joint between SCI and the CCRMA group. CCRMA stands for Center for Computer Research on Music and Acoustics, and it was started by two friends of mine, John Chowning and Leland Smith. This NSF grant, again, was way, way ahead of its time. The idea was put two microphones up and have a cellist and a violinist play a duet for 30 seconds. Now, the computer's going to munch on that acoustic data and spit out the score. Now, Mozart could do this for entire symphonies with a hundred players and the symphony went for 30 minutes. We found it very hard to do it for two instruments and, you know, 30 seconds of data. But, you know, you have to start somewhere.

Hsu: Yes.

McCune: My friend Joe Rockmore, who, starting in 1978 and all the way through 2013, was essentially my closest colleague, actually ran that project. CYAP was this program that Ed Feigenbaum brought over, and another friend of mine, Bob Drazovich, and Rowland Payne, ran that project. In 1978, I was still finishing my thesis.

Advanced Decision Systems

McCune: Then what happened is in August of 1979, the boss of this group, the boss of Cordell Green, left Systems Control, Inc. He came back and started talking to someone. He said, "I'm going to start a new company. We're going to do decision systems." That's artificial intelligence databases, control and estimation theory, pattern recognition, decision analysis, databases. It's all the different types of algorithms that you can automate in a computer to help people either to automatically make decisions or to help people make decisions. AI was starting to get hot, and this is 1979. But it wasn't a pure AI company. In fact, there were five founders: Bob Drazovich and Rowland Payne, Dick Wishner, the principal founder, myself, and then a guy named Edison Tse, who was a professor in the Engineering Economic Systems Department at Stanford. He didn't do AI at all, and Dick really didn't do AI. He did control systems, estimation systems.

We thought, once again, if you smash all these different technologies together and you start bringing in problems to solve, we will come up with innovative solutions. Turns out that I would say 70% or 80% of what we did was artificial intelligence, but not exclusively, and I think we were one of the few places that had experts in all these other fields.

Hsu: Right. This was Advanced Decision Systems.

McCune: Yes. In 1979, we started a company called AI & DS, Advanced Information & Decision Systems. People started calling it AIDS in 1980, and within a couple of years the AIDS epidemic started going. We realized that, in fact, people would see on my car I was from AIDS and I had a PhD. On a plane back from Washington they'd ask me about how the epidemic's going. I mean, it got pretty ludicrous, so we changed the name from Advanced Information & Decision Systems to just Advanced Decision Systems. We didn't do that until about 1984, I think, maybe, 1983, 1984.

Expert Systems in the 1980s

Hsu: One quick question.

McCune: Yes.

Hsu: During this time, while you're working on your dissertation and then at SCI and then starting ADS, did you understand your work to be part of expert systems?

McCune: Oh, that's a good question. Yes, that's a very good question. Just as now, in research fields there are different groups of interest and camps and wars, if you will. I mean, there are intellectual battles on what the right approach is. At Stanford we had two groups: Ed Feigenbaum's group, which was called Heuristic Programming Project and later Knowledge Systems Laboratory, and then we had the Stanford AI lab under John McCarthy. Early on in the 1960s, they were actually together, and then Ed, as he realized he wanted to do these expert systems, separated out.

Even though I was at the Stanford AI lab, where we were doing robotics, vision, speech understanding, natural language understanding, music generation and understanding, and program synthesis, if you looked at a lot of the technologies in there it really was, if not expert systems technology it was knowledge-based system technology. That is, it was the same technology, but there was knowledge in those systems; the knowledge was gleaned by the engine, the researchers themselves. They didn't go to someone else and say, "How should I understand this image of blocks stacked on a table?" They just figured it out from first principles, so I would say those were not expert systems. Those were knowledge-based systems, but the distinction's fairly minor.

So yes, I would say we were doing expert systems. This whole SY system really was attempting to encode the expertise of what it takes to describe an algorithm in English and turn out optimal code from it. It looks like that's really an expert system, except it's an expert system where we were the experts. We were some of the best programmers around, so we just intuited from our own experiences what the system should do.

Hsu: Right. You mentioned Ed Feigenbaum. You were working with Cordell Green, but not directly with Ed Feigenbaum.

McCune: Yes. That's right.

Hsu: What influence did Ed have on the work that you guys were doing?

McCune: Well, he gave talks. It was in the ether. Everyone knew about all these approaches, as we knew about Hearsay-II, which was actually being done at that time at Carnegie Mellon University. We knew about work at MIT, and the University of Edinburgh really was the fourth big center.

In those days, as today, AI can chew up lots of cycles. I mean, that's why we're suddenly seeing deep-learning work, because there's data and lots of computing power available. We didn't have that in the 1960s and 1970s, so DARPA said, "It's a strategic technology for the defense of the country during the Cold War, but we can't allow every university to have a $10 million computer, so we'll pick a few winners."

They gave Carnegie Mellon, MIT, and Stanford 10-year grants, multimillion-dollar grants every year, for 10 years, and they basically said, "Make us be the best in AI in the world." That's what happened. Of course, they picked the smartest people in the world, the guys that started the field in the 1950s: John McCarthy, Marvin Minsky, Allen Newell, and Herb Simon at Carnegie Mellon University. So yes, I would say certainly in program synthesis, we thought we were doing expert systems, and certainly when we got to doing SYAP at Systems Control, even though Ed was no longer involved, we were definitely doing an expert system.

They would actually go down and talk to what are called LOFAR gram analysts at Moffett Field, because they were flying these P-3s and looking for submarines off the coast of the United States. They were experts that knew how to find Soviet submarines. They actually went and talked to those experts. It wasn't even a classic expert system, because it was a hybrid. One of the reasons for using a Hearsay-II or blackboard model is you have a central blackboard where you post results or hypotheses and eventually what's left on that blackboard is the answer. Sort of like you had an equation and different students came up to the blackboard, started working on the equation. When you're all done, there's an answer. A lot of people contributed. There was only one blackboard. When you're all done, what's left is the answer.

Now, in our case, we have knowledge sources, and these knowledge sources can be expert systems. They can be database lookup. They can be sensors that are sending in data, interaction with a human. It's a very general problem-solving paradigm because you can integrate all sorts of data and all sorts of algorithms. That is a classic expert system for very hard problems. If you have simpler problems, using something like EMYCIN production rules is much better, and we'll get back to that because that's the way I went with my search engine. Those are the sort of the two problem-solving paradigms that I used essentially in my career.

Applied Research at ADS

Hsu: Getting back to ADS, was it a lot of the work that you did was for DARPA or the Defense Department? Was the company targeting that specifically? Was that just the business that was coming up?

McCune: Two answers. One, we were targeting it because we knew those people. I'd worked for DARPA since 1972 at Stanford. Then the SYAP thing was for DARPA.

Hsu: Right. Oh, because all your research in graduate school was being funded by DARPA.

McCune: Yes, yes. I mean, 95% of the AI lab was DARPA funding. But a more important reason is the following: There were expert system shell and expert system consulting companies sprouting up. We had all of them here yesterday at the expert systems workshop. Their goal was to sell their shell, sell their consulting, go out into corporate America and make money building expert systems for that.

That was not our goal. We didn't want to solve today's problem. We wanted to do applied research and prototyping to solve much, much harder problems. It wasn't basic research. We wanted to be driven by real problems, but it was more like the DARPA model. You know, there's basic research—go build me a low-risk application—and then there's all this stuff in between called applied research, advanced prototyping, proof of principle systems.

That's where we wanted to be, and actually, for the first five years, that's where we were. Then, some of those things started working. Then people said, "Well, build me a real system." The last half of the 1980s I started building operational prototypes and operational systems for the government. But we started by doing the research and building the prototypes and proving that the basic idea was sound. You know, if you can prove an idea is sound for $500,000, why spend $10 million trying to build it and maybe failing?

It's a nice phased approach to doing injection of new technologies, shall we say, into the world. We actually had a different business model. In fact, we were self-funded. We had no venture capital, no outside funders. There were five founders. We each put in, I think, $5,000 or $10,000 cash. I think later we had some problems. We each had to put in another few thousand, and that was it.

Then we started immediately getting contracts, and that paid our salaries and we kept growing. We went from five people to 200, 200 plus three spinoffs.

Hsu: Wow.

McCune: Well, that was over a 12-year period. But it started with nothing.

<u>ADS Radar Target Classification (RTC) System</u>

Hsu: We're going to start with the first project or the first client, I guess.

McCune: There were some studies that we were doing because often you're paid to do a study for six months and then they say, "That's a great idea." Study often is a paid proposal. "We'll give you $50,000. You write us a proposal for $5 million," and then with $50,000 you better know what you're doing. Then we'll say 'Yes' or 'No.'"

The first actual system we built was something called a radar target classification or RTC system. It internally looked exactly like SYAP and Hearsay-II, at least initially. It was a blackboard system. The idea was to take the same old AI architecture, actually new architecture, and apply it to the problem of understanding what are called ISAR, Inverted Synthetic Aperture Radar, images. This was a new technology, and it can look out over the horizon.

The idea was that the Navy would be able to put this new type of radar on the ships, and if there's something, someone, coming at them hundreds of miles away, you would see them before they could see and shoot at you basically. But it's a difficult problem because the images that come out are smeared. I can show you examples of them. You need, again, knowledge of different types to understand.

In the end, you boiled up all this information from the images as well as ancillary information. Then you had a library of all the different ships, and you matched your hypothesis of this image against a library of potentially all the ships, large ships in the world. You'd say, "Which ship is it?"

Now, there's a problem with that. By the way, that's called model-based because it's actually not just a library. They're actual models, 3D models, so you're matching the model of each ship type to the model that you have intuited from the data and just see which one matches. Sometimes they're very close. Sometimes it's like, "Well, I can get it down to one of these three ships" or classes of ships.

That's the first thing we did. That was proven to work. I'm not sure it was deployed because in those days, you know, it had to be built by someone and put on the ships, and we didn't do that. That was the first thing. The reason I mention it is, one, it was technically successful, and two, it was a direct follow-on from the SYAP work that Bob Drazovich and Rowland Payne had done. In fact, they ran this program, not me. I just helped in ancillary ways.

Blackboard Architectures

Hsu: You mentioned the blackboard architecture. I'm not very familiar with that. Could you explain that in just a couple of sentences?

McCune: Sure. It's basically a central data structure surrounded by a number of what are called knowledge sources, but they're basically separate algorithms with a very simple interface. These algorithms are allowed to look at data inside this blackboard and do some computation and then put new data back out. Now, the data, it starts out, is low-level data, but what you put back out is an intermediate-level hypothesis. Someone else looks at that intermediate-level hypothesis and says, "Here's a higher level."

So, you can work from something like an ISAR image but end up with a node that says, "This is a Soviet cruiser." It's basically a shared memory with a particular format that allows you to share the sorts of data you need to share.

Hsu: It's sort of like a collaborative form of collecting knowledge?

McCune: It's a collaborative form of managing the problem-solving process when you don't have a single algorithm that can solve the process. You need separate knowledge sources.

An easier example would be in Hearsay-II in speech. You have some knowledge sources that know about phonemes and some that know about words and some that know about sentences. Well, the sentence guy doesn't want to do anything until the word knowledge source has said, "Here's a word, and here's another word." Then the sentence guy can come and say, "Oh. That looks like..."

Hsu: Oh, okay. I see. You have things that are sort of layered or multilevel that depends on something lower. The lower thing works first, and then the next thing comes and takes the outputs of the lower things.

McCune: Right. But it can go the other way too. Someone could say, "You just said, 'I saw the [blank].'" You could say, "I think it's going to be, 'I saw the ball,'" because that's the sentence I'm expecting. If you go back down, look at the signal, can you tell me that the person actually said ball? Sometimes that top-down processing is faster than bottom-up, because it is goal directed, it's focused. This is a very, very general problem-solving architecture. It allows you to do bottom-up, top-down, or both bottom-up and top-down.

In fact, usually there's something called a meta-controller that controls which knowledge source is called upon next. This was because the computers were so slow and we didn't have parallel machines, so we couldn't just sort of do everything. You had to control the order in which these processors worked, because if you did it in the wrong order, one processor might be so slow that one knowledge source would never come back. It would never produce a result.

Now, I should mention that this is such a generic architecture that it's good for doing the R&D. It's actually poor for delivering a system, because it's inefficient. What you end up doing is finding out which of these control paths are actually used for different classes of problems and then somehow compiling that so that everything runs faster.

When you don't know the right control architecture, but you want to see if your algorithms are working, this is probably the most general architecture one can think of.

Associate for Science and Technology Analysis (ASTA) System

Hsu: Let's finish the signal stuff.

McCune: Okay. The next signal system was called ASTA, the Associate for Science and Technology Analysis. That's government speak. It really was a system to model and reverse-engineer radar systems. We mentioned before that the horizon radar was a big advantage, because you could see the enemy before they could see you. If you blow that up into multiple year increments, you want to know what the Soviets are going to deploy two years from now in terms of radar technology, because maybe they've got a better radar. If you actually got into a conflict and you got within 100 miles, which used to be far enough away that they couldn't see you, but now suddenly they can see you. They can also shoot you. It becomes really important in the Navy where they have these ships that are worth billions of dollars to know, “What does the other guy have?”

The way they do this is they intercept the radar signals that the enemy’s emitting, typically from a test platform on land or eventually a sea trial where they don’t build it permanently on the ship, but they just install it there temporarily. You can start to see that it’s a test radar essentially. You start intercepting the signal. You fly by and take photos of the actual antenna because now you can tell how big the aperture of it is and what type of radar it is, whether the antenna’s rotating or if it’s an all-digital radar. Then you might get information, like, someone might steal or hand you a copy of the radar manual.

There are all sorts of ways that this information can come in. Some of it’s textual, some of it’s signal intercept, some of it’s photograph. Immediately, in our case, you could say, “Well, you could do that in a Hearsay architecture,” but what we wanted to do was have a constraint-based system. So, we built it in the MRS language, which Mike Genesereth developed at Stanford University. He was a professor at Stanford, and he consulted for us on how to use his inference engine.

The key thing about his inference engine was that you could run inferences forward and backward. If someone said, “I know the radar has a certain pulse repetition rate,” you could put that data in and then let the system infer everything it could from that. Or if someone said, “I know that this radar is used for a certain purpose or as a certain aperture,” you put that in and let it ripple. It was a very flexible inference engine. We could’ve used Hearsay-II architecture, but we didn’t need to go quite to that level of generality.

Hsu: What do you mean by a constraint-based system?

McCune: Okay. A radar has a number of subsystems, and the goal is to tie down each of those subsystems, the design approach. There are only so many ways to build a klystron tube, so many ways to build an antenna. Now, if somebody came up with a brand-new one, we probably could not discover it because we weren’t doing learning. This is strictly a knowledge-based system. But the idea is if you could constrain one subsystem in the radar in a certain way, it might then constrain what you know about a different subsystem, and around you go.

As you get more information incrementally—and this process might take years, by the way—you put in each new piece of information and you see where you are, and hopefully, eventually, it pops out and says, “Oh, this is a blah-blah radar. We’ve seen it before,” or “We’ve been expecting it.” Or “It’s just like a radar that we’ve been testing.” That’s more typical.

You know, it’s a race. We know about a lot of things because our radar people are designing the same sort. Basically, this knowledge base was a knowledge base of design rules about how to design an arbitrary radar, and the goal was to feed it information and incrementally constrain the space of all radars down to, "It's this radar." That's why I call it a constraint-based system.

Hsu: I see. What was the advantage of this over the Hearsay style system?

McCune: Well, it's more efficient. If you can do it the way we did, it's just simply more efficient. Hearsay is the most general architecture. If you don't know anything better, that's probably where you want to start.

Hsu: But you know this is a constrained space.

McCune: Yes. We understood that it's fairly constrained space. I mean, a radar is a radar. It's pretty well understood, so we didn't think we were going to be seeing something or having a new source of data that would require a new knowledge source. It really just wasn't warranted to do the full Hearsay architecture.

We built that. We tested it out, and this is an interesting story that came up yesterday. We built it for the Navy. The Air Force found out about it and said, "We want it." Then the Army found out about it. Now, it's running in MRS, which is based in Lisp. At some point, they said, "Okay. We're done with the research. Deploy this thing."

Well, the Air Force, the Army, and the Navy each had different types of computers. They weren't going to change. None of the computers ran the Lisp language, so they asked for a quote. "How do you port this from Lisp to C?" or something worse. I don't even think it was C in those days.

Hsu: Were they using ADA by that point?

McCune: No. That was before ADA had rolled out. Yes, that's a good question. This was mid-1980s; I don't know, when ADA went live. Probably maybe late 1980s. We tried to convince them not to do that, but eventually we gave them a quote. I think at that point they said that it's too much money and it's not programmed. We needed to program that amount five years earlier, so the whole thing sort of blew up, which is sad, because they wanted it.

We'd proven it worked, but there were some budgetary and bureaucratic restraints, but also technical. That is, how do you get a sophisticated inference engine into practical use? That's still a problem today, because the language and the tools you use to do the research typically are not the ones you want to use to deploy such a complex system. I don't think that problem's really gone away. In this case, we failed. The government and ADS failed because we didn't get to go to the next step, which is unfortunate.

Advanced Signal Processing Computer

McCune: Then the next system was the Advanced Signal Processing computer. I can't say much about it because it's a classified system, but I want to get to the punch line because it's important for the context of the history of expert systems. The idea was there were these signals that were being collected, and there were rooms full of people with a little bit of computer front-end processing that were poring over these signals trying to figure out which were of interest. You know, maybe one-one-hundredth of 1% of all the signals every day are ones that some PhD needs to look at, do something further with. Everything else was sort of like garden vanilla, no big deal.

At some point when you have hundreds of people doing this, even though these people may be enlisted military people or hired contractors without PhDs, you're still spending a lot of money. So, they said, "Can you do anything to help automate the front end?"

That's what we did. We built a little expert system that did the front 75% of that job, and we built it for $1 million. Five or six years later I went back and they said, "So far you've saved us half a billion." That's a 500 to 1 return on investment. They were very happy, let me tell you. They just saved $499 million.

It's an example of where expert systems can play a role. If you have some expertise at the level of a doctor or a professor or a technical expert, you can bottle it up and hand it out to a lot of people. Like, you could take MYCIN, that does blood, infectious disease diagnosis, and give it to doctors or even nurses in Africa, which we could not do, unfortunately, in the 1970s. That would be an analogous win, right? But the idea of just creating MYCIN so that existing MDs can use it doesn't make any sense, because MDs already know all that.

Here's an example where you could actually get to a bottom-line return on investment calculation, which is what corporations like to do and even the Defense Department likes to do. This is an example of one of those systems with a real number attached to it. There are others in the Defense Department in the logistics area. Well, I take it back. Those don't have the number associated with. What they have is the speed and quality of the logistics. Like how do you move 300,000 soldiers in one month from here to Kuwait? That was a real problem before Desert Storm. They solved it, but you can't put a price tag on that. I mean, the real win is we won the war. Okay. Those are the signal analysis systems.

Hsu: You worked on these sort of in succession and then simultaneously with these other things?

McCune: Yes, that's right. RTC was in the early 1980s. AST was in the mid-1980s. ASPIC was in the mid-to-late 1980s. Then in parallel with all that, we did text understandings. I really had sort of two foci, if you will.

Text Understanding

Hsu: How did the text understanding stuff come about?

McCune: I already mentioned the News Service Systems was an expert user at sort of the state of the art. Then in 1981, the government said, “We’re getting more and more text in terms of these reports that our people write.”—user manuals for radars, email, and what they would call cable traffic coming from overseas bases and companies. Most of the world’s data really is in text. High-level data. Human processible data. They said, “What can you do?” We can look at the data that comes in, but if someone says a year later, “What happened a year ago?” we can never find the report or the message or the cable traffic on that.

I got to thinking about the News Service System, and I thought it really didn’t work because it was just keywords. Then I started thinking about MYCIN and EMYCIN, which is a rule-based system, production rules with weights on each rule. If you could process documents with a set of rules, with predefined weights, you could actually create a dynamic weight for each document with respect to the topic area or the concept area that that set of rules defined.

We got a contract and did a test of that, and it worked beautifully. What is called in the search space “precision and recall”—I can show you a graph—was like almost perfect for a small test case.

Hsu: These were the techniques that were done in MYCIN. Where had they been pioneered initially?

McCune: MYCIN was in the Knowledge Systems Laboratory. A guy named Ted Shortliffe worked in Ed’s lab. He didn’t actually work directly for Ed. He worked with Bruce Buchanan, who was Ed’s deputy. I knew that work, and then after Ted Shortliffe’s thesis in the early 1970s, a guy named Bill van Melle built EMYCIN, which was empty or essential MYCIN. That was the production rule engine with an editor that allowed you to put your own production rules in.

I thought, “Gee, if I had EMYCIN…” I mean, my notion was, “I’ll get rid of the News Service, keep the AP line coming in, and put EMYCIN on there and type in my own rules and suddenly everything’s going to be wonderful.” And it was. It worked.

Bill van Melle was one of my housemates at Stanford, so this gets into the sociology of Silicon Valley. We had this one house from 1974 to 1980, and it had three bedrooms. There were always three computer science students living in this house. The first three were there, and then some of them graduated, and Bill and Lyle Ramshaw and I moved in. I knew Bill, and I knew about his thesis. I knew about that and I knew about the News Service, so I came up with a system and amazingly it worked beautifully. We just kept working on it. We had to scale up the knowledge, I'm sorry, the database.

A guy named Vic Askman was a database guru, so he created an inverted file structure so we could store all the text files or the handle for those and then essentially index those by every keyword in the text. Then our EMYCIN would only go against this index file of all the keywords. Basically, you had every keyword that existed in any document, one or more documents, and for that keyword you had a list of the document numbers or handles that had that keyword in it. That's basically just an index. You never had to look at the documents again. I mean, almost all tech systems work that way.

We scaled up that, the back end. We created an editor so we could build the rules, and then one of the things we did is we started building operational systems for the government. One of the things we did is we put a front end on it so you could do a GIS, a geographical information system, a map, and you could say things like, "Is there any message traffic in the last 24 hours?" You could draw a polygon on a spot of the Pacific Ocean, any message traffic that says there's a noncombatant oil crew vessel in this polygon and do that search. The search is actually being done against text. What the user is doing is inscribing a polygon and giving a couple of concepts that he wants—like ships of a certain type—and then everything else is automated. So pretty cool.

Early Text Search and Query Technology

McCune: The other thing we did is a real-time scanner so that instead of just walking up to RUBRIC and saying, "I want all articles in the last 24 hours about Soviet cruisers," you could place a standing query. Actually, you and 200 people at an airbase or an intelligence center or company, which is what happened later, 10 years later. We'll get to it. Everyone could establish their standing queries in the form of one or more of these little rule bases. Then every time a new document came in, all of the rule bases would be fired against that one document. If one of them matched above a certain threshold, which is on a zero to one scale, you would get an email, an alert, essentially, on your computer, saying, "This document just came in."

We did that by, I don't know, 1985 roughly. There's this real-time dynamic version. There was retrospective retrieval. There was large document indexing, and then, of course, there was the knowledge creation and management of the rules themselves. We built all that and started applying it in many areas. Fabulously successful.

I want to mention that a knowledge base, in this case, a set of rules about a single topic or concept that you're looking for in the text, isn't really an expert system. Let's say I'm looking for, I don't know, Soviet cruisers. I didn't go talk to five experts in Washington, DC, and ask, "What do you look for when you're looking for Soviet cruisers?" No. I, the user, input my own rules. Again, I'm sort of my own expert, essentially, even though maybe I'm not. I came to call these systems preference systems instead of expert systems. The rules now are capturing my personal preferences for what documents I want to see.

Think about watching your sports teams. You don't need an expert on baseball to know what words are going to show up in tomorrow's box score on the San Francisco Giants. You could create that yourself. —

In fact, we did this. A guy named John Layman started a company to create these concept structures and sell them to people, so that everyone could get a high-quality knowledge base and not have to do all the work.

But the main thing I wanted to mention is, again, it's not a classic expert system. It's more like a personal preferential system. Let me also compare this technology and the philosophy behind it to things like the Google search engine. Because everyone's familiar with that and it's like, "Ah, Google, solve the problem." You know, "Go away." This predated Google by, like, 15 years.

Other than that. When the internet really took off in the 1990s, so 10 years later, the problem was there was a lot of text out there and you wanted to make the search engine really, really simple to use. Today you have a very simple engine. You just type in keywords and a bunch of documents immediately appear because they've indexed the world. That happens very fast. Thank you.

But for every query you give it, basically you get back an infinite number of documents prioritized in some weird way. It's not actually that weird, but it's not very good necessarily. You have to wade through. They're counting on the fact that you're like everyone else; when you put in these two words, you're going to want what everybody else wanted. But what if that's not true? If that's not true, you're sunk. You know, I use Google to do research on obscure topics. Doesn't work very well.

I was trying to do something different. My assumption was you have people, one or a small group of people, who are the best in the world. They don't want to just walk in and say, "What's the score of the Giants yesterday?" Every morning, they want to do what I used to do back at the AI lab. They want to see "Did we find any more," I don't know, "Soviet cruisers?" or "What happened to the stock price of my company on NASDAQ yesterday?" These are knowledge workers. They have more or less the same queries every day, and multiple people have the same query. That means you should spend more time building up the perfect query so that they get back one or two documents with the answer. They don't get back a list of, you know, 500 documents that they have to wade through.

It's really sort of the other end of the spectrum from what the Google search engine was designed to do and does very, very well. This really was designed for knowledge workers in corporations or large government labs or what have you, or maybe university departments that have the same damn query every day, but they want good answers.

Hsu: Wait. Who was the client for this?

McCune: The initial funding came from the intelligence community, but it spread. It was like ASTA. I mean, better than ASTA. Everybody. Every three-letter agency you've ever heard of and the Army and the Navy. Everyone wanted it, and then, you know, the State Department, the FBI. Then basically this thing really took off.

Verity Spin-Off

McCune: In 1986, in parallel with this, we had other technologies that I was not involved in at ADS. We had a list of like 20 technologies that we thought were ripe to productize in some way, either for the government or maybe with venture capital. RUBRIC was number two on the list, so we created a little group led by Cliff Reid, and it consisted of Cliff, Phil Nelson, Mike Cashian and two guys from MIT. Really crackerjack programmers, smart guys, and they tried to implement the first thing and it failed. Then they said, "Okay."

Hsu: What was the first thing?

McCune: I can't remember. I'll have to look that up or ask Cliff. They made a mistake, obviously. It's like information retrieval. The first thing's not necessarily the right article. They went to RUBRIC and built it, rebuilt it. Up to this point, in 1986, it was in Lisp. Very slow, but flexible. We could try out all sorts of things with it. They rebuilt it in C. It was lightning fast. We took in venture capitalists, and we hit the market at just the right time because the client/server architectures were taking off. Every company in the world was adding Sun workstations, the big databases, corporate databases on some server.

How do you get it the information? So, we spun off. We incorporated a company called Verity. Spun it off. We essentially sold Verity all of the RUBRIC technology as well as this Topic prototype that was done by Cliff and company.

Hsu: Topic was the C version?

McCune: Yes. I don't think it was called Topic then. I'm not sure. I'll call it Topic, but it effectively became the Topic product, even though it existed before the company existed.

Hsu: Oh, okay. What later was sold as Topic by Verity, the technology became that.

McCune: That's right. Yes. What we did for a while there, and actually after Verity was started, is we would go to a new client and say, "We're going to build you a customized engine. We'll build the back end to get the data in and index it. We'll build the knowledge bases and the graphical user interface, but the core engine, we're going to license, or you're eventually going to license from Verity for a set fee." That became the initial product. We set that company up with the intellectual property of RUBRIC and the Topic prototype. We also gave Verity seven full-time employees, including Cliff and Phil and Mike Cashian. Later Richard Tong. I did not go, because as I mentioned earlier, I hate to call it dabble, but I don't like to work in one area for a long time. That's why I was doing both signals and text. I mean, nobody's crazy enough to do that.

But I was. So, we took seven of our best people, and we raised venture capital. We brought in a guy who was on our board of directors, Mike Pliner, to be the CEO of Verity. They took off, and after a few years they were still using something too close to the model I mentioned of having this engine and a lot of development and consulting services around it. You couldn't go public with that, and people really wanted something more shrink-wrapped.

They didn't get rid of that, but we brought in a new CEO, Philippe Courtot. He came out with a PC shrink-wrapped version, sort of a low-end version of Topic. I can't remember what year that was. I think it was 1993 or 1994. At that time the internet, of course, was taking off. Verity went public in 1995. I think they sort of spun it as an internet company even though it really wasn't. It was a search company, but at that time there wasn't a Google, so nobody knew that that was a good thing or important.

It was the second largest Silicon Valley IPO of 1995. The first one was Netscape. Verity was number two. The development of the technology, the prototyping of the product, the creation of Verity and the going public was just perfectly timed for not only the technology, that it's a client/server. More text was getting put into databases by corporations. The internet was becoming available, so we just lucked out. What can I say?

Verity Growth and Sale to HP

McCune: Really. A lot of this is about luck. Within a year or two after going public in 1995, Verity had a market cap of $1.5 billion. It did very, very well, and then it moved into knowledge management and eventually was sold to another knowledge management company. Then that company eventually was sold to Hewlett-Packard. It's actually a product you can buy today from Hewlett-Packard. My friends at Hewlett-Packard say it's still the largest search engine in the world when measured by number of seat licenses sold.

Obviously, you don't buy a Google search license. What you get from them is advertising. So, it's a totally different business model. This was a classic buy site license, buy a license for a thousand people in your company or whatever. Classic early software, enterprise software sale. Yes, that's about it.

I want to mention the people who made RUBRIC happen besides me. Jeff Dean and I really were the coinventors or cocreators of the whole concept, and Jeff and Dan Shapiro and later Richard Tong and Vic Askman, who I mentioned before, were the implementers of it. We did a lot of research to figure out better ways of doing things, and John Layman built a lot of these applications and spun off this company to do the knowledge engineering of these concept bases. Then a little bit later Joe Rockmore took over this whole area for me. I sort of moved up a level, and he became the head of all the text work. As I mentioned, I had been working with him since 1978, since I went to Systems Control, Inc.

Additional ADS Spin-Offs

Hsu: I think you mentioned yesterday that there were two other spinouts of ADS. Successful companies?

McCune: Yes. There was one in a biotechnology workstation that spun out. Both of them spun out after 1988. I think it spun out in 1990, and eventually failed. I don't know because I'd moved on by then, so I don't know exactly why that was.

In between there was a company called Aerial, which I was involved with to some degree. It was wonderful technology. Let's call it a belief network, also known as a Bayesian network. It was an editor, and it allowed you to create belief networks, compute them, and then most importantly visualize them. You could actually view a belief network and the data in the network as a spreadsheet or as a graph structure. I think they had one other way you could view it, and that was wonderful technology.

I think it would have succeeded, but it was not VC funded. It was funded by a company that was trying to pivot called Mad Computer Corporation. Mad Computer Corporation in 1989 had the world's fastest but also most expensive PC. Turns out that's not what corporate America wanted. They wanted really cheap PCs where they could buy 10,000 at once, so Mad eventually went out of business. Along the way, they stopped funding Aeriel. We couldn't pivot fast enough to get that level of funding, so it went under.

So, we did three spin-offs, one of which was wildly successful, and the other two failed, but not for good reasons, I think. As I said, a lot of this is just timing and luck.

ADS Sale to Booz Allen Hamilton

Hsu: Then ADS continues to chug along on its own until 1991?

McCune: Yes. We spun out Verity in 1988. Everyone said, “Oh, this is a really great idea.” That’s why we spun out two more companies. The problem was there were another six people, mid-level managers, that wanted to be queued up so we could spin out their company, so the core R&D group lost cohesion. Basically, it became a big management nightmare of how to keep the R&D group that we’d started in 1979 going, generating ideas, but also bring in venture capital and spin people out.

I think it could have been a sustainable model, but in our particular instance, which is actually one of the few in the world, it did not work. We could see the R&D company itself was not continuing to grow, so we decided to sell the company. We’d done three spinoffs, and they hadn’t succeeded or failed yet at that time, so we sold the company to Booz Allen Hamilton and became a division of Booz Allen Hamilton.

At that point, I got a promotion. I became the chief technologist in essence of Booz Allen Hamilton, which was a worldwide consulting company which I think, in those days, had maybe $600 million in revenue. Now it’s a multibillion-dollar revenue company.

Hsu: Why them?

McCune: They did system engineering work, but they weren’t really software developers. They could do simple integration of simple systems, mostly relational databases. To me, it was pretty low-tech stuff, but also to some of their visionaries, it was low-tech stuff, so they wanted to get into system integration. They could see that companies like Computer Sciences Corporation and SAIC and IBM, of course, and others were making more money on larger contracts. Instead of a $500,000 contract, they were looking to make it a $5 million or a $50 million contract, so they bought us to get technology and to get some expertise at how to build these systems and how to convince clients to hire us.

By then, we had some expertise in this commercialization process, but we had also installed a number of these RUBRIC Topic systems as well as Aspic and others. We had technology and some expertise at system building, so that’s why they bought us.

The experience didn’t work out very well because we looked more like an SRI International, highly paid think tank, and they were a very buttoned-down, let’s go consultant for IBM or the State Department. It was a massive cultural mismatch, so most of the middle managers from ADS that moved over were gone within a year and a half. In fact, I was gone within just over a year. I’d say within two years, there were only a dozen senior people, either senior technologists or managers left at Booz Allen, so your question is a very good one.

We had a good story, but what we didn’t understand was much more important than, what do I bring to the table and what do they bring to the table? Do those things complement each other?

The cultures didn’t complement each other. Business schools teach case studies. A typical merger and acquisition does not work, and it’s not because everybody doesn’t want to make money or they all don’t have the same vision, it’s that the cultures don’t match and the people that are in the acquired company can’t get along. They can’t figure out how to fit into the new culture.

I mean, it’s simple things like at Booz Allen you had to have a Mont Blanc pen that cost $100. For somebody coming out of Stanford University, Silicon Valley, the notion that that’s important to you or your bosses or your employees or your clients is so foreign as to be ludicrous. That’s just one tiny example, but there are hundreds of things like that. So, it really didn’t work, and within a year, I left.

Hsu: Well, what had you been doing? What did you say your role there was?

McCune: I was trying to help them. There were like 50 practices. It was a very flat organization. Each practice was headed by a partner. They were scattered all around the world, and I was trying to help maybe half of them learn about how to do software development and integration and learn about a variety of technologies. I mean, AI was really just one, so it was very basic stuff.

The best relational databases, client/server architectures, you name it—we had a lecture series on all these different topics. We had cross-fertilization where ADS or other practices would give talks about what they were doing so everybody else could see that. It was a coming-up-to-speed exercise, which would have been fine, except for the cultural issues.

<u>Montgomery and Associates</u>

Hsu: Then you leave, and you start up a new Montgomery and Associates office.

McCune: Montgomery and Associates was a little consulting company with maybe a dozen people headquartered in Santa Monica, California. They had one guy, a very senior guy who eventually became the Assistant Secretary of the Air Force, Larry Delaney. He ran the office in Washington, DC.

I started an office for them on Sand Hill Road, and we were doing more business consulting for technology industries, that was mergers and acquisitions, and things like, “Tell me the size of this market in five years,” so market assessments, all sorts of stuff. They used my mini-MBA experience. I had spent a few weeks at the Stanford Business School in one of their short courses, and of course I had by now more than a decade of experience of my own in high tech, so we were basically going out and helping various companies do various things.

Hsu: You were taking this short course while you were at ADS?

McCune: Yes. That's right.: Yes. They paid two or three of us to go back to become mini-MBAs. As we got bigger, we realized we needed more management expertise, and we can bring it in, which we did, but we can also all become smarter. Really good idea.

Hsu: That lasted for two years until 1995?

McCune: Right. That was just over two years. That's right.

Hsu: It was during this time that you did the work for the FAA [Federal Aviation Administration]?

McCune: Yes. As I said, we did all sorts of different types of work for the government and aerospace companies. We did work for the Jet Propulsion Laboratory and Livermore National Labs. Because of my background in advanced software, I got hooked up with a group at the Center for Naval Analyses, and they were asked by the director of the FAA to analyze this large system that IBM was building for them called the Advanced Automation System, AAS. It was going to be the next iteration of air traffic control. We actually have it now 25 years later.

I'm not sure how much detail you want to go into, but it was the largest software build in the history of the federal government, probably still is. The original budget for it was $1.5 billion. We came in and looked at it, had them re-cost it, and this is how I ran software. It's a lot of hardware scattered around all the airports in the country, and we figured it was going to end up being north of $5 billion, so we recommended ("we" meaning this large committee, and I was just on the software subcommittee) that they potentially cancel the current contract and break it up into smaller pieces, recompete it. That's what they did.

Hsu: You also have this Peace Shield lawsuit.

McCune: Yes. One of the things I did at Montgomery and then also later at Cyladian is expert witness work. On much larger programs, there is no one expert. For example, in Peace Shield, Boeing had been fired, terminated for cause, terminated for default. That is, the Air Force said, "You're building this big radar system for us, and you haven't met a milestone. We've decided to just cancel the contract, and under the terms of the contract, the half a billion dollars we owe you, we're not paying you." Boeing said, "But your termination terms are incorrect."

We went in and said we'll play outside experts as if we were the government. We're going to look at all your hardware and software and where things stood on the day of termination and prove to a court that there were no grounds for the termination. Even if the government doesn't want to start it back up, they owe you all the money you were owed on that day, and we did that.

Hsu: I have a couple other things I wanted to ask about. It says on your resume you were a panelist on the 1993 Technology Summit along with the Secretary of Defense William Perry and Under Secretary of Commerce Mary Good.

McCune: Yes. Bill Clinton got elected and came into office in I guess January of 1993. One of the first things he wanted to do was inject high technology as a way to grow the economy, so he held this summit near San Francisco International Airport. Al Gore shared the summit, and they had all sorts of undersecretaries and assistant secretaries and experts on this, that, and the other thing.

By the way, 1993 was after 1989. You'll recall that in 1989 the Berlin Wall fell, and by 1993, the Soviet Union had collapsed, so what was happening was the federal government was drawing down the budgets of the defense, the national security complex. You're going to have hundreds of thousands of people, some of whom have PhDs. What are you going to do with these people? One panel had to do with reinvesting the seed corn that we paid a lot of money for during the Cold War, and how do you get those people out into commercial industry? That's what that panel was all about. That's why you had an Under Secretary of Commerce as well as Bill Perry, who was I think at the time the Deputy Secretary of Defense. Then he moved up a little bit later.

I was on there because I had done what we've talked about. ADS was primarily funded by the Defense Department, and we had spun out a few companies, one of which was Verity. That became very successful, so I had a little bit of history in high tech and also making defense-related technologies into commercial products.

Defense Applications at ADS

Hsu: I wanted to step back really quick to ask about two systems that were used in military operations, one in the invasion of Panama and then in Operation Desert Storm. Were these related to the stuff at ADS that you had mentioned?

McCune: They were ADS systems. I developed them, or my group developed them, but we haven't talked about them.

Hsu: Oh, I see. Okay.

McCune: You want to hear about them? The one in Panama really ended up being a piece of hardware, so it's sort of irrelevant. It was someone else's hardware, and we reverseengineered it. Suddenly, the Pentagon said we're going to invade, I guess, Haiti or Dominica. I think it was Haiti.

Hsu: First George Bush invaded Panama to remove Manuel Noriega in Operation Just Cause?

McCune: Oh, Panama. Okay. I don't know about that one. I think we were involved in that too, but I can't remember how.

Hsu: I guess Reagan had invaded Grenada or something.

McCune: Grenada. Maybe that was it.

Hsu: That was an earlier one.

McCune: But it was not AI. It was just some ancillary thing to a project we had been working on.

Hsu: The one in Desert Storm was an expert system?

McCune: Yes. That was a Bayesian network for learning on Scud missile launches. It was developed to give us a warning if the Soviet Union started getting ready. As you may know, you can't just push a button and launch a missile. You have to fuel the missile, or in the case of these Scuds, they're launches off of trucks that launch from the back. The trucks are parked, and they have to have gasoline. The missiles have to have a warhead, especially if it's a nuclear warhead. That makes a big difference, so it's nice to know where the warhead came from. We had a system that would, if you had the right sensors, would figure all this out, what was going on, on these bases throughout Eastern Europe.

The Cold War ended, but suddenly, the Iraqis had Scuds, and they were launching them at anyplace they could—the Saudis, Kuwait, Israel; they were launching all over the place. We used this system to help alert when they were.

It didn't quite work right though because it was knowledge-based. The nice thing about the Soviet Union is they always do everything the same way. They had rigorous training, and they never deviated. Even though you give the Scud missile to the Iraqis, you have no idea how they're going to use it. They don't have the training, the money. They don't probably have the same launchers, so this is a very good example of if you build an expert system and you make it work really, really well on this one problem, at the end you say, "Oh, it's just a Scud missile launch. Bring it in here." You have one day to turn it on. It's not going to work because the knowledge base isn't right anymore. It was used, and it worked to some degree, but of course, it could have been better.

Cyladian

Hsu: We have very few, very little time left. Maybe we could talk really quickly about your work at Cyladian?

McCune: Montgomery Associates changed strategic directions, and a lot of partners left. By then, I had hired Joe Rockmore at Montgomery, so we both left and at the same time started Cyladian. Cyladian existed for I guess 18 years. It was always just the two of us, so we got rid of all the management problems. The only problem we had is market; do our consulting services work and make clients happy? That's what we did for 18 years.

It worked very similar to what we've been talking about, except I now no longer had a development team and sometimes DARPA brought me in. They had three or four contractors, and they had put the software system together to be installed at Pearl Harbor, but they couldn't get it to work, so they made me the manager for a year. I got that project back on track because it's something I sort of knew how to do, having done it many, many times.

Hsu: So just sort of consulting on different projects, coming in.

McCune: Yes. All sorts of different projects. We did more legal work. We got very good at breaking software patents. We can talk about that. My claim is that if you give me a software patent, I can break any software patent because there's always prior art, and I can find it.

Hsu: Right, meaning invalidate it.

McCune: Invalidate the patent. We only did it maybe three times, but to me, it seems pretty easy. Basically, the patent system, as you probably know, in this country's broke, and this is just an indication of it. The people up front will give someone a patent for anything, and then you have to fight it in court to show that they should never have had that patent in the first place. That's really what was going on here. Some of these patents were worth $1 billion. We did some work for Nintendo. They had a new videogame coming out, but they were infringing somebody's patent, so we had to show that the patent was invalidated so they could get their product out. I mean, things like that.

Hsu: You also created the stock index.

McCune: Yes. You may remember, prior to 2000, there was a bubble and a big crash, a tech crash in 2000. Prior to that time, the internet and Silicon Valley were going crazy, and the stock market was going crazy. There were a lot of high-tech magazines started: *Wired* magazine, *Upside*, and three or four others.

Upside hired me as a consultant. They were opening an office in Times Square, and they were going to have an internet radio station about high-tech companies. They said, “We want to track all of the companies as they go public. We want our own stock index. We don’t want the NASDAQ 100 or the S&P 500,” so I created for them the Upside 150. It had 150 high-tech stocks, and it didn’t matter which main market they were on. It was a matrix, and all 150 companies fit into one slot in this 2D matrix. One dimension of the matrix was hardware, computing level hardware, software, and componentry, chips and boards and stuff. I can’t remember what the other matrix was. Each company was in one of these cells, so let’s say it was 3x5. That’s 15, so 15 different numbers and then sort of a composite number for the whole 150. That way you could see if software companies were going up while hardware companies were not.

That was an interesting way to keep tabs on the public high-tech market, and *USA Today* said this is the best technology index there is. Unfortunately, it was only operational for a couple years because, after 2000, most of these magazines went under.

It was actually a lot of fun doing that, and it was way beyond the S&P 500 index and others that are out there, even the ones that are handmade. Some magazine would say, “I just picked 20 companies, and that’s my technology index,” but why those 20 companies? “Oh, I don't know.” I had a knowledge-based organization of the technology space that is the information technology space.

Hsu: To sort of wrap up, at Cyladian, you went back to doing a lot of contracts for DARPA again.

McCune: Oh, yes.

Hsu: Was there any particular reason? You knew the space, or you liked being in the defense space?

McCune: I think maybe I got a little bit bored with all these consulting gigs I’ve been talking about, and I had friends that were rotating in as program managers at DARPA. Dick Wushner started a whole new office at DARPA, and he hired Joe and me to help him.

We were hiring program managers, starting new programs, helping them manage the programs. We were like program managers except we didn’t work for the government, and we didn’t have their authority either, but we could basically be jacks of all trade and do a little bit of everything. I started around a dozen programs, some in AI, some in cybersecurity.

We tried to automate the social sciences, a wide range of subjects. I think I mentioned a few of them in there: large knowledge bases, computer security, learning by instruction was a neat one. We realized that these learning algorithms are never really going to work. What you want to do is create a knowledge base and a learning algorithm that interacts with an expert and is essentially taught by the expert. The expert's now a teacher, and the teacher could say, we're going to learn algebra and I'm going to teach you what the plus sign means and the equal sign, how to solve algebraic problems. The teacher would actually talk to the computer that way, and the computer would build up a symbolic knowledge base not in their own head, not a Bayesian net, but actually symbolic knowledge that can be used to explain and solve other problems, which I think in the long run is the right way to do learning. That's what people are good at. That's why you go to school for K-12. That's the way you learn knowledge incrementally, so we tried to build a computer and had reasonable success doing that, but again, that's really at the hard end of learning.

Hsu: Finishing up, you stopped working at Cyladian in 2013?

McCune: Yes. Basically, my wife died in 2011. By the end of 2012, I said I'm going to stop doing what I'm doing because I don't feel I'm giving enough attention for value added to my clients. Joe went off and kept doing his thing with another company, and I retired in December of 2012. We actually shut the company down at the end of 2013, and I've been retired ever since and not thinking about any of these sorts of things until this week when Ed Feigenbaum called me and said we're reviewing expert systems from 1955 to 1990. I go, "Oh my god." I'm amazed I've remembered anything from way back then.

Final Thoughts

Hsu: What is the thing that you're most proud of in your career?

McCune: Well, at one level, it's that I've been able multiple times to walk into a new situation with just my wealth of experience, but no knowledge of this new company or domain or problem, and quickly get up to speed and help them solve a very valuable problem. I have multiple instances where I either made a company a billion dollars, saved them a billion dollars, avoided a lawsuit for a billion dollars, I mean, large amounts of money. I really wish I had been on a commission—you know, a 1% commission over the years. I would be a very wealthy person now.

I think my basic idea, going back to high school, is that I'm a problem-solver. I'm essentially a consulting engineer by mental predilection, interests, style, and I got to help a lot of people. In the middle of that, I had my own company, and I actually got to develop my own software, so I think I probably had the most fun and the biggest impact doing that. That was ADS, that 12-year period. That was probably the most interesting part of my career.

Hsu: Where do you think the computer industry and maybe in particular AI is headed in the future?

McCune: I think we're going to see a lot more cars running over a lot more women. I've been predicting this for years. It finally happened, so this gets back to the tension between expert and knowledge-based systems, general-purpose systems, learning by neural nets versus learning by example or learning by explicit instruction. Everything that we're doing today in deep learning is neural net based. It's totally data-driven. If the data doesn't cover all the edge-cases, the one in a million-time cases, and that woman walks across the street, not at a crosswalk, you will run the woman over because that's the nature of that learning algorithm.

Now, they're starting to put expert systems on top to control the learning algorithms. That's a very good idea, so I think we're ending up back to this Hearsay-II model. We're going to have hybrid approaches with multiple learning algorithms, multiple expert systems, other algorithms that are not expert systems at all that do automated feature recognition out of images or whatever. You bring all that expertise together, and it's like bringing 10 experts on different topics in the room and handing them a problem. They're going to do better than any one expert. That's where we're headed. In a way, we're coming full circle back to the Hearsay-II model of the 1970s, which is good. There's hope. We're going to have problems before we have solutions, but we will have solutions.

Hsu: Lastly, what advice would you give to a young person starting out in their career today in the computer industry?

McCune: Well, I think you need to do what I did fairly early on, although I didn't do it explicitly. I didn't think I wanted to be a consultant, but I've always wanted to help other people. I like to do that more than, say, be the CEO and build an empire so everyone knows I'm in charge of a company with a thousand people. To me, that's so much drudge work.

You need to know yourself, and the only way you can do that is by trying out different things: different areas of technology in college or high school, different programming assignments, different types of jobs, different types of bosses.

Don't get trapped too early because you can get trapped in being a programmer of this class of learning algorithm, and you can do it for 40 years. You can make tons of money, especially if you're good at it and it's the right area or algorithm, but you may be bored out of your mind. You may not think that the problems you're solving, or your company is solving, are important problems any longer, so don't get trapped. Keep an open mind.

Hsu: Thank you very much.

END OF THE INTERVIEW