Computing Educators Oral History Project

An Interview with *Mordechai (Moti) Ben-Ari*

Conducted Tuesday, July 1, 2008
At Madrid, Spain
Interview Conducted by Elizabeth

[0:00]

Elizabeth Adams: This is an interview with Moti Ben Ari from the Weizmann Institute in Israel conducted by Elizabeth Adams. This interview is being recorded on July 1st in Madrid in Spain. It is part of the Computing Educators Oral History Project.

Did I pronounce your name reasonably well?

Moti Ben-Ari: More or less.

E: More or less. [laughs] Okay. Can we start out by asking you something about your parents? Background, their work life, their education?

M: My father was a political scientist, grew up in Washington, D.C., worked for the State Department for a long time, and then retired and did some university teaching. My mother studied nutrition, had some jobs, and also took care of us.

E: Good thing to do.

M: Yes.

E: So neither of them were really in computer-related fields, mathematics?

M: Not at all. There is nothing of that in the family.

E: OK. What kind of student were you?

M: I was probably what you would call a geek or something like that, but, you know … the usual, got good grades and studied, didn't cause any problems.

E: Did you take any in math and science?

M: Yes,

E: Did you go to a special …

M: No. Public school.

E: Public school. And you had siblings?

M: I had one brother, two years younger.

E: Uh huh. Who followed in your footsteps?

M: No. He's an economist. So he makes more money.

E: Ah! [laughs] Did you all … both of you go to college?

M: Oh, yes.

E: What made you… I know your original degree was in mathematics at … MIT.

M: Yes.

E: And then from there you … changed your life?

M: It actually started before then, I had got … when I graduated high school I got a summer job writing programs for a scientist at a local university in FORTRAN on the big 7094s or whatever they were, and even though I was studying mathematics all the time, when I studied, computers were taught in the electrical engineering department. And I wasn't there, so I didn't touch a computer at school. But I did have these summer jobs and later on we have to go all through the different turns of fortune, but this was always a decent way to make a living. I always say that computers and programming are the refuge of second-class mathematicians.

E. [laughs]

M: And then eventually at one point I started continuing my studies.

E: Uh huh. And that was not in the United States?

M: No, then I had already moved to Israel, in 1972. I think it was 1974, I was working as a systems programmer at the university, and I started doing a master's degree in parallel, in computer science. That actually got me… (do you want me to start talking about … okay). That actually got me started into computer science education because when I started the degree they asked me … they gave me a list of courses because my degree was in mathematics, the courses I would have to take in computers. And then a few weeks before the semester was supposed to start they said, "Why don't you teach the operating systems course?" because I was a systems programmer and knew something about it. And then what happened was, of course, it's a mainframe computer, you can't have forty students changing the operating system. So I happened upon Dijkstra's original coordinating sequential processes in the book (I forget what it was called) and started teaching that stuff. And it worked pretty well and at the same time, or roughly at the same time, the first Pascal compiler came out and I convinced the university to fork over 160 Swiss francs for the tape.

[4:38]

E: Wow.

M: And it wasn't that much. Yes. And there was the Pascal-S interpreter there and I had the idea of turning this into a concurrent interpreter. And then one … by that time, no it was probably, maybe a bit later, I had started a doctorate and my advisor went away for the summer, which is what all advisors did then, and so I wrote up my notes that I had written for the course I was teaching in operating systems into a book on concurrent programming. And I sent it off to a few publishers and … I think … very quickly I got a telegram from Henry Hirschberg who was then the big, one of the big bosses at Prentice-Hall, saying, "Tony Hoare likes your book."

E: Wow! [laughing]

M: [chuckling] Well, I almost, you know, fell over from that. In the end he didn't like it enough to put it in his red-and-white series at the time, that came later.

E: Right, I remember that.

M: They still decided to publish it. And when you're the first person on the block, it doesn't really matter what you do. You corner the market, which I did for quite a long time.

E: Right, so this is after … you'd written this while you were working on your master's?

M: And doctorate.

E: And doctorate, okay. And those were both from Tel Aviv University? So … why did you choose MIT? Just to go back a little bit.

M: Because you were a good student and you were interested in science and I didn't get into Harvard and so I got into MIT. And that's a sore point to this day, probably.

E: [chuckling] And you knew you wanted to major in math at that point.

M: Yes.

E: I mean, we didn't have computer science.

M: I'd still like to major in math.

E: Oh, okay, good. Good. Did you enjoy the research you did for your computer science doctorate? (Enjoy is a funny word.)

M: Well it depends. I actually switched advisors twice because of people going on sabbaticals, I … let's see if I get this right. I started out with Amir Pnueli and switched to Zvi Galil. And then I started with Zvi Galil and I was one lemma from the end of proving something reasonable, reasonably important, which I didn't, so then I switched topics to Amir Pnueli. And I really liked the work with the concurrency, with the logic. These are the subjects that still interest me forty-six years after I wrote my first program.

[7:25]

E: Oh, that's interesting. And the mathematics helped, in that your mathematical background was helpful in that area … ?

M; Yes, I did theoretical work for my master's and doctoral thesis. And I still think mathematics is a good thing to major in because I found later on, that when I started working in industry, that anytime something came up, it was really just a bunch of equations. And when you have the mathematical background and you're not afraid of going and looking at the literature, then you can get into everything much faster than you can otherwise.

E: That makes sense. I noticed from your vita that you had some visiting experiences as a professor. You went to various places.

M: Well, professor's much later on.

E: Okay … so take me through that career development if that's good.

M: Well, I've probably never really been a good research-oriented academic, I think. One thing I've noticed, for example, Amir Pnueli, Turing prize winner and all that, he's working on the same thing he invented thirty years ago. And I keep saying I have an attention deficiency disorder because I can't work on any project more than a year. Well, maybe three years. So I'm really not the type to be a specialist researcher in something. And that's always changed every few years and I don't remember exactly the circumstances, but as I was finishing my doctorate I was offered a very good position at an Israeli research and defense institution and I decided to go there. And it was really good because it gave me a chance to do real sort of things. Usually I was working on trying to suggest to people how to write better software, whether it is trying to convince people that bubble sort is not the best way of doing a sort in a real time system or you could write a real time system in a high-level language, not in assembler, introducing the first software engineering procedures … and at a certain point actually managing people in software development groups, at which I was less successful.

E: [chuckling] Well, I notice that you have down that you were also … that was the manager of software development at Orbit Systems.

M: Oh, that was just a few months.

E: Just a few months, and then you said …

M: That's old. I don't even mention that any more.

E: Okay, well, we don't have to. And … so you … let me think for a minute. Okay, so that takes us up to about nineteen-eighty …

M: No, about 1990 or so, nineteen-ninety-something … And then I had a few short jobs and I tried to start my own start-up writing a program … no point going into it. It was supposed to be educational software, but for various reasons … really, it's not worth going into.

E: Okay.

M: And then in 1995 I found out about the Department of Science Teaching at the Weizmann Institute and was accepted there. Fortunately, the department head, Uri Ganiel, took a chance on me and agreed to finance me until I could go through the whole academic appointment procedure. And that turned out to be extremely successful because looking back, well, I sometimes say that the only thing I know how to do is write good textbooks and write software, educational software, and I was always doing that on the side. You know, I'd think, "This is something that people need," and I'd start to write a textbook that is … what we call in Hebrew a "*chaltura*," a moonlighting … and here I was sort of being paid full-time to do the only thing I know how to do.

E: Uh huh. Well, it's good. You have certainly been successful at it. So you have actually been at the Weizmann since 1995. That's your association, even though you've gone other places … twice.

M: Well, beforehand, once during the industrial phase, we actually had a sabbatical … a possibility of sabbatical. I spent a year at Brandeis and I wrote a textbook in logic, which is not what I was supposed to do but that's what I did. And then, during this time at the Weizmann Institute, I took two half-sabbaticals in Finland. And this was a very interesting situation. I had come up with the idea of using animation to … this goes back to the mental models that you have of a computer. I think you always have to understand one level below what you're working. So if you're working at a programming level, you have to understand something about assembler and architecture. You maybe don't have to understand gates and quantum mechanics, but one level below you have to. And I thought visualization and animation would be a good idea for this. So I went to, I think it was PPIG, in 1998. I met there Erkki Sutinen, who was at the University of Helsinki there, and he said, "Oh well, we've been doing this. He has this Eliot or Jeliot system, I think is was Jeliot by then. "And sure, you can have it." So I looked at it and it was very good, but it was intended for advanced students learning sophisticated string algorithms. And it was just too heavy for beginning students. So what I said is, "Well, maybe I'll make a few changes." And he sends me the source and all the variable names and comments are in Finnish, which I didn't know then at all.

[13:59]

E: But you do now.

M: Now I know a bit. So I had them ship me over a summer student to make the few changes. And of course, he's a brilliant student and all that, and actually he's also a good artist, and so he doesn't like the design and he doesn't like the code and he doesn't like anything and he's going to start from the beginning. So fortunately he managed, before he got fed up with it, managed to come up with a working system. And that started the long-term work with the Jeliot and the collaboration with the Finns. And, coincidentally, a year after I met Erkki and started working with him I met Anita [now my wife], who is originally from Finland, and we started going together. And so now I have my own private translator for any trips to Finland.

E: Oh that's always useful. And she travels with you a lot now, I see. Lovely person.

You've had a number of … you've been active with the visualization workshops … ?

M: Yes, at the beginning I was more.

E: You were very active with that. I noticed you were active in the Ada community for a while.

M: This I've come back to, actually. Well, this is another thing I could go back to. Sometimes I come across as very extreme, in the … being anti-something and for something. This goes back a long, long time, to the early 1980s probably, where I saw two cases where people worked for three or four weeks on bugs and then when I saw the Pascal tape sat on my desk and I started learning it and I said, "These bugs would not have been … would have been compilation errors in Pascal." One of them was … FORTRAN was passing everything by reference and a floating-point constant was passed by reference and changed. There's no way you can debug this.

The other one was a large commercial PL/I program. I actually worked in the insurance company once where it was illegal to use floating point. But because they tried to be nice to you, somebody tried to write *bit* and wrote *bin*, which is short for *floating binary*. Now that can happen, but as far as PL/I is concerned, it makes sense to automatically convert a floating-point value to a bit. So when I see entire companies running around themselves for three or four weeks trying to find a bug and I see this alternate approach, where this is a compilation error, then I became very pro-extreme-Pascal/Ada, these sort of things. And I probably would have made more money if I'd been an expert in C and C++, but I just can't do that. In fact, the last industrial work I ever did, I personally had to debug for two or three days because of an inconsistency in the declaration, the definition of a C function.

I still think Ada is really good. It's actually not dead as people think it is. And I'm re-writing my Ada textbook, which is the best textbook I ever wrote by the way, now with the help of Edmond Schonberg, from NYU and AdaCore.

E: Right, oh good.

M: I probably won't make much money off this but if I can save a few people from a few bugs and a few airplanes from crashing because of C type bugs, then …

E: And election systems from tallying things incorrectly, yes.

Oh! I notice you have a lot of … you have three PhD students in the resume that I saw. Noa …

M: Which resume is that?

E: It's probably out-of-date. Noa, Cecile, Yifat …

M: Noa [Ragonis], Cecile [Yehezkel], Yifat [Ben-David Kolikant], and now Ronit [Ben-Bassat Levy] is … next year she'll finish her Ph.D. And I'm also supervising by remote control Niko Myller of Joensu, Finland. We're waiting for his article to be accepted and that will wrap up his thesis.

E: Oh very good. And you have a number of master's students.

M: Yes, yes.

E: So, do you have a philosophy of teaching? What courses do you now teach? Or do you …

M: Well, we don't have any undergraduates so we don't have to teach. But I'm pretty good, I think, as a advisor. Well, mostly I'm hands-off which is why I wasn't such a great manager. Because to be a good manager you tell somebody to do something in a week, you expect it to be done in a week or to find a complaint why it wasn't done, an excuse why it wasn't done, and that doesn't work in industrial management, where you have to nag people all the time. But I work with students, I think the only students who work successfully with me are the people that if you say, "Well, do this in two weeks," it's done. And … really, my sort of approach is that … well, I have a new master's student now. She's always coming to me looking for answers, and she doesn't get answers. Because I like them to work independently. I say, "Do whatever you want," but when they come back then they are in for a very hard time until they can convince me that they know what they want and why they want it and what are the reasons. So I'm … you know, very nice and easy-going on one side and then very strict on the other side, especially when it comes to writing, because this is a sensitive issue for me. So, we have this expression "*sidrat g'nouch*", I don't know how to translate it … a sort of boot camp approach. That anybody who starts writing something, it takes a while.

[20:18]

E: Yes, I can see that.

M: But I've been very lucky in the quality of my students.

E: Well, probably your reputation makes it clear that students who have an independent ability and really want to work come to you. I'm sure that's the case.

You got the SIGCSE Award for Outstanding Contributions [to Computer Science Education]. I was lucky enough to hear your talk, "Why the Concorde Doesn't Fly Anymore."

M; Yes, something like that.

E: You have lots and lots of publications, that's a good thing, some with your students and some with your own.

M: No, all with my students.

E: All with your students! And I think I counted something like eight books so you really …

M: Probably closer to twelve.

E: Oh my. All right …

M: I've been working in … I came back to concurrency a while ago. We had developed a course in concurrent programming for the high schools with Yifat, my student ¥ifat Ben-David Kolikant, and it was so successful it was cancelled, which is the way that education ministries work. But this got me into it again and then again one of those serendipitous sort of things.

A few years ago, I started getting e-mails from Pieter Hartel, who's a professor at the University of Twente in the Netherlands. And he had started using my textbook but he wanted the students to work with Spin, the Spin model checker [http://spinroot.com/spin/whatispin.html]. Now this goes back a long way. Amir Pnueli and Zohar Manna had always been on the deductive-proof approach to verifying concurrent programs and, of course, I worked in this sort of thing, axiomatic systems. So the idea of model checking, just sort of brute-force checking all of the possibilities, was something I had a little bit of trepidation from or something like that. But eventually after whatever number of e-mails I got from Pieter, I started looking this, into Spin, and became totally addicted to it.

The reason is that with Spin, I have the best of both worlds, which you almost never see in computer science education.

On the one hand, this is a award-winning, fully professional-level software tool that's used in many, many places by people who put a lot of money and risk into it. And on the other hand, it's so simple you can teach it to beginning students. I could even show parts of it, I'm going to try next year, probably, to show it to high school students. So this really got me into it and I rewrote my concurrency textbook to be … not Spin-oriented, but Spin-friendly, and started developing various software tools. And then finally last year I published a textbook on Spin itself. So I'm sort of back into concurrency and this is one of the things I'm looking at now. I don't know if I want to publicize this yet, but what I'm also looking into, or started working on a bit, is trying to do a easy-to-understand re-implementation of Spin, where I'm not trying to get the best performance you need for the professional use, but the tool itself will be even more friendly to use and to extend and to understand the algorithms and so on.

E: Is Spin an acronym or a just a name?

M: No, it's just a name. Oh no, I think it is an acronym somehow or other. You'll have to look it up. The "P" is probably protocols. It was initially designed for a communications system.

E: And I notice that I've heard your name associated with constructivism.

M; Oh … oh yes.

E: Is that … shall we just … ?

M: No, no, that's fine. In 1995 I joined the Department of Science Teaching, which means I'm now working in education, I'm not … no longer working in computer science but computer science is, of course, the reason you're there. You're good at this and you do the transition into this. So I sat in on a few classes and started reading some things and the mantra there is *constructivism*. So, I looked into it and read ten … some articles and tried to apply it to computer science. And what I wrote then is okay, as far as it goes, in the sense that it raises issues that I think are still valid and it gives ways of … or it suggests ways of looking at issues that are still valid. But the problem with constructivism after I read thirty papers, I started getting more into what this really is and I started becoming part of those philosophers, especially [Malcolm] Knowles of New Zealand, those type of people, philosophers of science and education, who say, "Constructivism is either meaningless or trivial." And, well, it's either meaningless in the sense that it's related to philosophical doctrines like idealism, non-realistic philosophical doctrines that nobody really accepts unless you're a post-modernist or something like that. And its pedagogical claims, that students should be active in constructing knowledge, is something that is almost trivial. Everybody knows that if they do exercises in labs and talk together it's better than just hearing the lecture. So I haven't really pursued this. The only thing that this did lead to was an interest in the philosophy and history of science and science teaching and I actually wrote a book on this.

[26:51]

E: Sounds like anything you get interested in, you then write a book on.

M: Well, that’s what I know how to do.

E: [chuckling]

M: If I could publish big important research papers, I would do that. But this is what I can do.

E: Were there any particular challenges you found in your work environment, juggling commitments at work and at home?

M: No, nothing in particular.

E: Not for you. Did you have to make compromises in the course of your career or that you felt that you had to?

M: Yes, but this is the personal sort of thing that, like I said, if I had persevered in any one subject I would probably be a bigger expert. That is, if I had … you know … I worked on logic for a few years. I worked on programming languages. I worked on software engineering. I worked on actually building real-time systems. I worked on the history and philosophy of science and worked on visualization and worked on concurrency. So I … work on too many things but I’m a …

E: … a renaissance computer scientist.

M: Yeah, probably. Actually, at one point it did a … it’s an interesting anecdote, that at one point when I was working in the industry, one of the big system engineers would always ask for me to help him with the computer systems. And the reason is if he went to some real specialist they would see their little specialty, and I would always see more of the big picture. So there is a niche for people like us.

E: Right, right Do you have any strong outside interests that would help us (I’m not sure you haven’t already answered this), but that would enable us to understand you better or that have had a shaping effect on your career?

M: Maybe. I’m very interested in language as such. At one point I even started doing a master’s degree in linguistics and it was just a matter of a [financial] scholarship that somebody held too long, that didn’t come to me as it should, that made the difference between … and I had to go start working and I went to work in computers. If that scholarship had turned up I’d probably be a … you know, in linguistics now or something like that. The other thing that’s not really relevant, but I’m waiting to retire so I can be a historian.

E: Ah … [chuckling] If you could give a … Well, I guess the only thing we haven’t talked about is your activities in professional organizations. You have been active, you’ve …

M: Not really. I’m not a organization type person, and …

E: But you were on the task force for curriculum …

M: No, I was asked to do one little thing and I did it. If I'm asked I do things … you know, I do referees for all the conferences and journals. I'm the only person who always returns a journal or referee report within a week. And then I get …

E: You better be careful or they are going to send you more! [chuckling]

M: They do. In *Computer Science Education* I'm on the Editorial Board and then when my student … I was waiting six months for a review, it drives me crazy. Um, but I haven't really done very much.

E: Oh, well, you've been very successful without that. So, if you could give advice to … our question says "a young woman starting out," what would it be?

M: I actually wrote something, "Advice for a young woman." Maybe I'll try and publish it sometime. Um, well, I think this is very unfortunate that more women aren't going into computing because you only have to look what happens sort of down the road and it makes a big difference if you have your own capabilities and you're more independent and not dependent on other people. It actually sort of … well this is sort of personal, and I don't know if we want to leave this in, but my ex-wife was a physicist. So when you get divorced from somebody who has a career, you don't have to pay as much. But on the other hand, for the … for people themselves, if you look at 30% of marriages are dissolved and it's better if you have a good profession. You have to look at this even if you're not thrilled. But it's much better than, you know, being forced to go to court to get alimony and be a waitress or something like this. And there is … and computing is one of the more flexible positions. That, if you're an airline pilot and … or maybe not, but if you work in a factory or whatever, you're determined by shift work and things like that. Whereas with computing you're much more able to change jobs to go into different types of jobs, to be flexible, telecompute or whatever. And the people I've seen have really been very successful at this.

[31:55]

E: Sounds like that's a … something you should publish. I think it's good advice.

M: Okay.

E: Well, I guess if you could change one decision that you made along your career path, would you change? Or … ?

M: Probably almost all of them.

E: [laughing] Okay.

M: No, I'll have to think about this some more.

E: And is there one story you want to tell, that you remember … that as an ending to the interview, that you'd like to have remembered?

M: I'll have to think about this too.

E: Okay, well we can pick that up later. I think we've covered everything in our question list. I thank you very much for your time and it was fun for me to talk to you, so let me stop this.

M; What are the two things I'm supposed to think about?
[33:00]
[Break of several minutes, reconvene to continue the interview]

E: This is Part II of an interview with Mordechai (Moti) Ben Ari from the Weizmann Institute in Israel, conducted by Elizabeth Adams. It is being recorded on July 1st in Madrid, Spain and it is part of the Computing Educators Oral History Project.

Okay, Moti, I think you were going to tell me the advice you had to give young people. And I'm ready to listen.

M: Okay, I want to start with a story here. That I once worked for someone, a real-time project manager, and he said something that's really very hurtful in a sense. He said, "It's easier to teach computing to physicists than to teach physics to computer science graduates." And uh … there is some truth to what he said. And I also found this out later on when I was looking into something. I gave a talk once called "The Invisible Programmer." You can see it on the site. And that's that what people see is software on their PCs – like packages and the Internet – and they don't see that, in my opinion, most, that really most of programming goes on in hidden places where you don't see it. This goes on, of course, in airplanes, cars. I've recently seen a slide showing the new Mercedes car. It has 40 different programmable controllers and 600,000 lines of code. But this is not something that the students see and, on the other hand, it is of extreme importance for safety and reliability and so on. And what I even found, I even found an ad, I think by the Daimler Company, where they were looking for a software developer but they weren't looking for a computer science graduate. They were looking for a physicist or an engineer. And one of the problems is, I think, that a lot of computer science has become … well, of course, too technologically oriented, learning the latest tools and all that instead of the scientific principles. This you may remember from my SIGCSE address. But it's also become very introspective. That is, we place a lot of emphasis on writing tools for other computer scientists.

And I think if I had a piece of advice to give to students that if you really want to do something fun during your professional career, the best thing to do is learn computer science and something else. For example: computer science and physics, or computer science and biology, or computer science and finance. Because then you're in the really best position. On the one hand, you're not just playing around with tools, which is okay, but it's not going to get you very far. But once you start understanding the business you're in, then you very quickly get to very interesting positions where you understand what's being done and you have to come up with the computer solutions to do this. So if there's really something that I would say it's try to be interested in something else beyond the very basics of the computers themselves.

And at the same time, don't worry too much about whatever particular subjects you take in computer science as long as you cover all the basics. Because I think I said before, but if I didn't I'll say it now. Learning mathematics was very good because in any subject I went into, they threw a bunch of equations at you. So, it doesn't really matter if you've learned them or not, you can start learning them right now. The same thing here. If you're going to start working on options trading or control or … I worked once on something called control for flight control … Okay, I worked once on flight control. So they are equations and you learn them. And, if you know the basis of computers, if you know about architecture and you know about languages and you know about algorithms, if you know about protocols and databases, it doesn't really matter if you don't know the latest thing. And I think one of the big problems in computer science these days is the pressure from students and parents and industry that if you're not right away able to work with a certain tool, you're in bad shape. But my advice is don't worry about that. As long as you know your math, you know your science, you know your whatever, economics or biology, and you know all the basics of computer science, you'll do very well and you'll find lots of interesting things to do.

E: That's … sounds like good advice.

M: Okay.

E: Sounds like good advice. Do you want to comment for this interview about object-oriented programming at all?

M: Not really at this point. The only thing I want to say is that this has to do with the same sort of thing. That there is part of it now is rhetoric, but there is a hegemony of object-oriented. If you don't learn C something – C++, C#, Java – in your first year, then in all your years of college you're going to be in trouble. But I actually saw a figure saying that 95% of programming has to do with embedded controllers. I was never able to track down a real source, so I don't use this very often, but there's too much one-sidedness in the computer curriculum, too much on software packages and Internet programming and something like that. And there's an awful lot of interesting stuff that can be done where this isn't appropriate. If you're working for Daimler writing a … I saw an ad for somebody to write software for an automatic transmission. It sounds fascinating, but I don't think object-oriented programming is what they're going to use there. You need to know physics. You need to know engineering. You need architecture, algorithms and all the other things I mentioned.

E: I agree.

M: Okay.

E: Not so important …

M: You're not supposed to agree

E: I'm allowed, I'm allowed. Um … okay, did we … Is there anything you think we've left out, that you would like to have as part of your oral history?

[39:22]

M: Did we talk about model checking?

E: We talked a little bit about model checking and your involvement with it. I don't know if …

M: Well I'll just add something. You can probably edit it and put it in the right place

E: Or you can.

M: Okay, well I just want to say … let's say it this way. I think that formal methods are extremely important and it's true that not everything can be formally verified, either deductively or with model checking, but I have a slogan I once invented for myself, that once you learn this sort of thing you're a better programmer. And why is this? Because when once you start writing some code, I try to think … suppose somebody pulled a gun on you and said, "Your verification or your life!" Would you be able to verify it? And what I'm saying is that it makes me a more defensive programmer. I don't try to do clever things because I know that I can get in trouble and I'd never be able to verify it. So maybe I don't verify everything or almost anything, but it improves the way I do the program. And so again, I think this is another … again, I'm sort of extreme, nobody likes verification any more, but it's coming on and I'm actually very pleased with the fact that hardware isn't making any progress, though they have all these dual cores and quad cores and things like that. So that now people … now we can see downstairs that Intel has this big multi-core educational program and people are going to get in trouble and then they're going to start learning all the things we've been working on for thirty years or so. And so I think that's going to be exciting and is a good thing to make sure … this is more important to learn than the last package in the Java library or whatever. That you can learn as you need it.

E: To have the right mind set for …

M: Yes, to have the right mind set to learn the principles. I'll quote again what I did in the SIGCSE address and that was that I compared the computer science curriculum with the curriculum that my son took as a mechanical engineer. And we're very worried if you don't give them the right professional tools and projects in their first year. And if you look what these "backward" engineers do, they spend two-and-a-half years learning math, science, and the scientific principles of engineering. Then they start letting them learn about some nice application areas like robotics or whatever, and in their senior year they finally get to build a project. And I'm sort of disappointed that computer science education is not going in this direction. And it's too training-oriented and not enough into science and engineering.

E: I assume in that statement that we have quotes around the backwards in the engineering.

M: Yes, of course.

E: I want to make sure we're able to put that in.

M: They're … mechanical engineers are considered low-tech and software engineers high-tech for some reason. But strangely enough when I talk to my son, even though he's got a degree in engineering, he spends all of his time in front of the computer, of course, designing things, running simulations on control algorithms, things like that. Which relates to this, I've seen a lot. People don't want to learn computers because they don't want to spend their lives in front of computers. If you look, everybody spends their life in front of computers: insurance clerks and engineers and reservations people and pilots, for all that. They're just looking at all these computer screens. So everybody is doing it and you might as well be the person that understands it enough, the person that does that.

E: I think that's a very good point. Okay, it seems like I haven't got anything else to ask you, so I'll ask you one more time if there's anything else you want to add.

M: No, I think I've said enough.

E: Okay. So we'll close down the interview and again let me thank you for your participation.

[43:38]