An Interview with

DICK HEDGER

OH 378

Conducted by Philip L. Frana

on

17 May 2001

Minneapolis, Minnesota

Dick Hedger Interview

17 May 2001

Oral History 378

Abstract

Richard Hedger begins the interview describing his education in electrical engineering at the University of Minnesota. Following graduation he accepted a position at IBM Rochester in the optical character recognition group. He then discusses his shift to software development. Eventually he joined the Service and Support group developing application software supporting facilities infrastructures for the IBM System/3, and System/360. In various places in the interview, he describes some of the programming techniques in vogue while he was at IBM. He discusses his work in software for the System/32, System/38, and client server applications (attaching PCs to the AS/400). He talks about standards certification, e.g., ISO 9000. IBM people he discusses include Glenn Henry, Watts Humphrey, Ben Persons, and David Schleicher.

This is an Oral History with Dick Hedger on May 17, 2001, at the Charles Babbage Institute in Minneapolis, Minnesota for the Software History Project.

Frana: Thanks for coming Dick. I was wondering if we could start by having you describe your early education here at the University of Minnesota, your undergraduate, graduate courses and especially those courses, those activities, that you found most useful to your career when you left the university.

Hedger: Let's see. I started in September of 1957, and I chose electrical engineering as my major at the end of probably my freshman year. Mainly because I had a friend who was a little bit older than I was who was in electrical engineering and that looked interesting. So I went through five years of undergraduate education. I graduated in 1962 with a Bachelor of Electrical Engineering. The coursework and the things as I recall… we were the last class to have electronics, or have vacuum tubes. But there was quite a bit of theory mixed in with the design courses, but I thought what we learned was how to solve problems.

Frana: And did you know at the time that vacuum tubes were sort of an obsolete something to be studied?

Hedger: Oh yeah, we quickly got the transistors 077 and 073. Quickly got the transistors. Yes, that was only the very first quarter class I guess in electronics, basically. Yes, they were on their way out. So I joined IDM (IBM Rochester?) in June of 1962. And I'd already accepted an offer as an electrical engineer to work, I think in their optical character recognition group, that was starting in Rochester at the time. The lab had just started in early 1962, so I was in the first bunch of engineers hired into the development lab. Manufacturing had been going on five or six years, but development was done somewhere else. The way I got into software was, they called, and again called and wanted to know if I wanted to come down and interview the comp lab, the computation lab. Comp labs in those days were an IBM supported development lab, so they were an MIS facility that supported development labs. And I said, 'Oh sure.' So between the time I graduated and started work, I had a week or two off, and I said, 'Yes, I'll come down.' So I went down to Rochester and interviewed them and took the famous **DEPAT?** Test, and I thought, well, they wouldn't offer me that job. So lo and behold, they asked me what I wanted to do when I got done with the interviews, and I said, 'Well either the engineering job or the software job looks interesting. I'd take that one too.' So that's how I got in to software. I'd never had a programming class.

Frana: To that point?

Hedger: To that point. Never had a programming class. In fact, there weren't all that many offered.

Frana: Yes, now John Tukey coins the word software, I think it's '58, and then I know that I've talked to Walter Bauer who founded one of the early software companies called

Informatics, and he said that until oh ’63 or so they used to call it “proprietary programming items.” When did you first hear of the word software as a term that was used by lots of people? When did that sort of become ubiquitous, or did they call it programming for a very long time?

Hedger: Oh, it was clearly called programming. We were programming computers. And I think the word software came in to distinguish it from the hardware side.

Frana: And then ‘software engineering’ is a term that comes even later?

Hedger: Well, the ‘software engineering’ term comes later in probably in the early 1970s with various people trying to make it much more of an engineering discipline.

Frana: Okay. And could you sort of survey what is software? Does that change quite a bit over the 1960s and ‘70s and into the ‘80s? The idea of, I mean, is it this package thing? Is it something tangible? Or is it code at some point? Or does this idea of software really mean stuff that’s been put on disks for particular systems?

Hedger: Software is, I think, at least to myself and to what I did, software was either applications, pieces of code that solved problems, (or systems?). In my early career it was applications that solved what some engineer wanted or what some kind of system problems were trying to improve some process to do engineering. And then, for example, System 3 started up, and we had early, because we were trying to do the hardware and software in parallel, we had hardware simulators for the machines so the programmers could get started early, we had **some???** that run on large machines. And we built configuration management systems that ran on large machines. They had terminals that never had to have keypunches or do cards in those days, which was kind of, it danced and did an early version of System 3, system 360 NDS? Multi-tasking systems, early releases of the stuff and try to make it run and make it reliable. But then we got in to products, so then you thought of it as a product when you got into product development. So it was like the operating system was like a product that IBM sold. And it's not just the software. It's the documentation and the training and the help and all the things that go along with making it a successful product. And we treated it in a system. So we're not just selling the hardware or the software separately, we're selling a system made up of these products: hardware, computer itself, various I/O devices, operating systems, compilers, utilities, each one of those are separate.

Frana: It really does settle some questions. We had a software history conference last fall out in the Silicon Valley, out at Xerox PARC, and a lot of people grappled with this problem of, it was a conference on unbundling, and the idea of what is software and at what point is it a real product, was something that everybody really struggled with.

Hedger: Yes. See IBM unbundled the price of the operating system from the hardware in about what, 1969?

Frana: Yes. That's right.

Hedger: So then any software became a product. So you begin to think of them in product terms and you attached making money, monetary value to them.

Frana: And before you attached what? Service?

Hedger: Service and support…but again, the system idea there. We are selling a system to solve a customer's problem. So an example of how unbundling helped, affected us, at least personally, we had written a set of software which we called Multiple Terminal Monitor Task. It essentially supported multiple 2260 displays, where software engineers, programmers, on the System 3 could go and update their source code, provide configuration management, send their job off to a compiler to be compiled and assembled and downloaded. It was reasonably successful and from the very beginning a System 3 programmer never saw a card punch, which was late '60s was a little bit rare I think. We got that program well enough documented that IBM, which had a library at the time, I think that they were called Class 3 programs, included it in its library. You would put the program in the library and if enough customers were interested, the customers could order it for nothing.

Frana: And if enough customers were interested?

Hedger: No no, customers could order for nothing and no fee, they'd get some maintenance, they wouldn't get much support either, unless they called and you wanted to help. And we grew about twenty-five, thirty outside IBM customers who were sort of interested in this package. Well, and along came unbundling. And neatly everything had a price on it. And they sort of analyzed that whole set of programs and decided whether business wise they were going to do a lot of revenue or not. Of course we got shut down. So that's how unbundling personally affected me. [laughter]

Frana: So some programs were just eliminated as unprofitable?

Hedger: Yes. Whether they could have been in the future or not…I mean…

Frana: Hard to say.

Hedger: Yes. But there were very few programs in that library that as I recall actually made it to be real products.

Frana: Oh, okay. Do you know what percentage? I mean do you have any sense of that?

Hedger: No, I don't have a sense. But it was a fairly thick book of things that you could order.

Frana: It was called a **Class 3** programming library?

Hedger: Yes, there was **Class 1** which were regular products, **Class 2** were a different kind, **Class 3** were these, I think they were called **Class 3**, I'll have to look.
Yes they were.

Frana: Now, we talked a little bit at supper about IBM corporate culture and how now everyday is a casual Friday, more or less. There's a new book out , I don't know if you've read it, Po Bronson's *Nudist on the Lateshift*. And it talks about this nude programmer who exists sort of in myth out in the Silicon Valley. Was there even in the '60s a different culture among the programmers than the hardware people, or were they mostly alike at that point?

Hedger: Oh, they were probably alike.

Frana: And when did that big blue suit…?

Hedger: Through the early '70s it became much more casual. Though I guess managers kind of always wore maybe just a sport coat and a tie.

Frana: Do you recall major culture problems between the hardware and the software people, you know, or inabilities to recognize the problems that the other side faced?

Hedger: In my opinion, in the early days of the System/38, the hardware guys were always ahead of the software guys. And so they were always, 'Why can't you guys get it done faster?' But designing something or building something from scratch, a large piece of software, growing a team, doing all those things that went in to an original 38 is very hard, and in a lot of cases the engineers, hardware engineers didn't understand why it takes so long. And especially when we needed some new hardware features in the middle of it.

Frana: So they'd bring a new piece of hardware in?

Hedger: No, we actually had to add a bit to memory so the security system would work. About the second pass, the third pass of the design. But we decided that was important so we went back and did it.

Frana: I'm jumping ahead a little bit here, but as you read Fred Brooks's book on *The Mythical Man Month*, was there a similar problem at IBM Rochester? He talks about the hardware people talking about why can't the software people, you know, meet their deadlines? I guess Brooks' law is that, it doesn't matter how many programmers you throw at it, the more you throw at it, it's still going to be…

Hedger: Oh, how does it go? Nine people can't make a baby in a month.

Frana: Right.

Hedger: Right. And it grows exponentially in difficulty because of all the communication that has to happen as the size of the organization grows. So as you're building an organization it probably started out at about 90 or 100 and ends up at about 450. There are lots of growing pains over a period of time to get that done.

Frana: So it's a scaling problem?

Hedger: It's a scaling problem. And a communication problem in there, and a consistency problem.

Frana: And in Rochester, was it difficult to find people to work? Were they recruited from the West Coast, the East Coast?

Hedger: Primarily recruited from midwestern sources. When various parts of IBM would change their mission we would often go recruit in those sites, so we recruited internally at IBM as well. Like when our government, federal systems division operation changed its mission we got people from Grand Forks when Safeguard shut down.

Frana: Oh, when Project Safeguard shut down? That's interesting.

Hedger: Sure. And when the air traffic control system moved from IBM to Unisys in early 1974, '75 from Atlantic City. People who had lived in the Midwest wanted to come back. So they were usually not hard to recruit.

Frana: And you'd said at supper that people that came from the East Coast came reluctantly but didn't want to leave?

Hedger: Yes, usually they liked Rochester. Yes, didn't want to leave.

Frana: If you were going to identify the universities that contributed the most employees, would it be University of Minnesota first?

Hedger: Minnesota…well, I don't know if I can in rank them in size, but University of Minnesota, University of Wisconsin, Iowa State, some from Nebraska, North Dakota State University, a significant number from there.

Frana: Has that changed over time at all?

Hedger: The thing that's changed over time is computer science has become broader in the schools. Now IBM goes to places like Mankato State, Winona State, University of Wisconsin LaCrosse, schools here I can't think of the name, Eau Claire, in Wisconsin, there's another one in there. So it's much broader to a larger set of students.

Frana: Okay. Did you ever have to do, it seems anyone who has ever been an IBM employee has had to do some form of recruitment. Were you sent out to some of these schools to recruit?

Hedger: Yes. Oh yes.

Frana: And what kinds of people were you looking for in the ‘70s.

Hedger: By that time we were looking for people who had computer science or programming in their background.

Frana: Okay. At what point do you start getting large numbers of computer science majors working at IBM Rochester?

Hedger: Probably middle 1970s. Early 1970s. There was still a lot of changing of disciplines, and you trained them to be programmers. They’d come with aptitude and you, like myself, you train them to be a programmer. So there was a lot of inside training inside IBM as well.

Frana: Is there an IBM type? Is there something that distinguishes, I mean are you looking for people that seem to be family people or are you looking for raw aptitude?

Hedger: We’re looking for that and then if they came with some experience, depending on what kind of skills you were looking for, like database or transaction processing, compilers, or operating systems or you’d look for someone with interest and skill, it’s kind of specialty to fit the needs you had which you were trying to develop.

Frana: Getting back to the list a little bit here, I was wondering if for the record here, if you could describe how we get from the 360 System to AS/400 at IBM Rochester, or System 3, I guess at Rochester.

Hedger: Well, System/3 started out, this is the way I recall it, as a mid-range system with the idea we were going to sell to customers that had little or no computing experience, and in fact, probably didn't have a programmer on their staff. So we were aimed at the small business where we were going to essentially automate what had been this punch card business. And so the idea was started around, from a software center, started around RPG and simple business applications.

Frana: And then, now you worked primarily on System /38?

Hedger: I provided support facilities for the infrastructure for System/3.

Frana: But there's also a System/36, and we talked a little bit on the phone about this.

Hedger: Yes, so the System/3 grew and was successful and added transaction processing, we were able to **hit???** terminals and pretty soon you could do sixteen and thirty two and as these businesses grew, both their batch processing and their on-line transaction processing grew and memory size and processor speed and all that sort of stuff kept up. The System/32 was, I believe, a stand-alone system for a single operator. With the same, basically the same architecture, hardware architecture as the System/3, the same instruction set and RPG based. But so it was kind of a low end entry, it became the entry into the product line back in the early 1970s.

Frana: So it replaced System/3 in a sense?

Hedger: Low-end. Yes. As the System/3 became de rigueur for customers, this guy kind of slipped in. The electronics was different. It had the first high integration chips.

Frana: Oh, integrated circuits?

Hedger: Integrated circuits. And then it grew, well we can grow these add terminals and communications to them, and pretty soon we had transaction processing going on in those, too. But it was architecturally limited by how the operating system had been built and the kinds of how it could grow in scale over time.

Frana: All right. So it couldn't scale in the same way that the System/3 could?

Hedger: No, not as the System/38 could later on. It had architectural limitations.

Frana: Okay. And so System/38 is devised to address that need?

Hedger: System/38 was devised as a replacement system for the System/3.

Frana: For System/3?

Hedger: For System/3. That was our first target, to replace those guys.

Frana: So there was a direct line between System/3 and System/38?

Hedger: Yes, move those customers to the System/38.

Frana: I read somewhere that that was perhaps the largest programming effort in IBM at that time. It was just an enormous programming effort.

Hedger: I don't think it was the largest in IBM, but it was the largest in the division we were in, which was ? I'm sure OS/360 was bigger and larger.

Frana: Okay. All right.

Hedger: Yes, yes, yes. [laughter]

Frana: Is that something you have to say in order to keep the mystique of the OS/360?

Hedger: Well we never wanted to tell them how big the operating system was on the AS/ 400, because we were these intermediate small systems guys, and if we were building an operating **???** system they all looked kind of funny at us, what are you software guys doing? [laughter]

Frana: How large was System/38 when it was…do you have …?

Hedger: First release was a million lines of code.

Frana: A million lines?

Hedger: Yes. Between the vertical microcode, the operating system and the compilers.

Frana: And what kind of time period are we talking about?

Hedger: It started well, 1973-1974, and shipped the first one in 1980.

Frana: Okay. I'm going to be talking to a few of your friends. Old colleagues down at IBM: Ben Persons, Dave Schleicher, Glenn Henry, Don van Ryn, and Roger Taylor. Were they working with you on System/38?

Hedger: Yes.

Frana: Were some of these men working on different projects? Could you sort of describe these people and what your relationship was with them at the time?

Hedger: Glenn Henry was the program manager. Manager of the programming group.

Frana: Right. And we know he's ended up now in Austin, Texas.

Hedger: Austin, Texas. Dave Schleicher came in as the manager in the vertical microcode and later became *the* manager of the vertical microcode. Roger Taylor was an early architect and designer and managed the operating system for a while. And then Ben Persons came and managed the operating system **???** Burke for a while. Ben Persons came from the 34/36 area. And Don van Ryn was the programming manager of the 32/34/36 area.

Frana: Okay. And did all of you work on AS/400?

Hedger: Glenn Henry didn't work on AS/400.

Frana: Because he left too soon?

Hedger: He left for Austin.

Frana: Now what defines a mid-range system? Is there a size? How do you define, you know it's not…

Hedger: Price.

Frana: It boils down to price?

Hedger: Yes, I have to think for a minute. I think it was price. I mean they started at somewhat less than?, but this was kind of before PC days, so they started kind of, as I recall, on the order of $50,000 and it would grow to several million.

Frana: It's funny that you should mention that. I don't recall now where I read this, but I remember reading somewhere that what IBM liked to do was quote a price and stick to that price, but vary the service and give them the system based on the price that they had agreed upon with the customer, and sort of go from there. So that the customer never had this surprise, this shock, to discover that it would cost a lot more than they originally had contracted for.

Hedger: I was never in on the marketing side. That was another world to me.

Frana: Once it rolled out?

Hedger: Once it rolled out it was another world, yes.Yes.

Frana: Okay. That’s fine. Well, now I know, I don’t know if it was in your resume, you talk about the Malcolm Baldrige Award and how important that was to your systems development group to win this award. What is the Malcolm Baldrige Award for?

Hedger: The Malcolm Baldrige Award is a quality award given by the United States Government Department of Commerce for producing, based on a set of criteria for high quality products and services. There are, I think three classes: one is for services, one is for small business, and one is for large business. I think it is fairly prestigious and it’s in a response to the Demming awards in Japan, and there are several others. The site, the Rochester Site won the Baldrige Award, primarily for work on AS/400. But the roots of that were set up back in the development of the 38. In the software side. Essentially what we learned in getting the System/38 done, relative to discipline or process and metrics and measurements and how to manage teams. All the things you learn in developing the first releases of things. Brand new products, brand new system.

Frana: And there’s a relationship between that and ISO 9000?

Hedger: Yes, ISO 9000 is more of a kind of certification kind of thing. The process to get it is different. I guess anybody that puts their mind to it can get ISO certified. Whereas, the Malcolm Baldrige Award is somewhat of a competition that you participate in against a set of criteria.

Frana: I see. So ISO 9000 is sort of a standard?

Hedger: Standard right. It’s a TQM standard that you can get yourself assessed to and say yes.

Frana: Right. I got you.

Hedger: Their process is, they do what they say they are doing.

Frana: Okay. I went by the Army munitions plant north of town and there’s a big sign up saying they are now ISO certified or something.

Hedger: Right. Right.

Frana: So it’s all sorts of different areas.

Hedger: It’s all sorts of different areas. It can be software, it can be hardware. It primarily started, grew out of manufacturing, grew out of Europe.

Frana: Okay. Now you mentioned Demming? And TQM. Is MDQ, is that an IBM version of TQM, and are they similar?

Hedger: MDQ? MDQ?

Frana: It’s sort of a software quality standard. I’m not sure where I came across that acronym, but it’s total quality management, is that what we are talking about here, with the Malcolm Baldrige Award?

Hedger: Yes, and various awards. Rochester Software, rather the AS/400 product got ISO certified after the Malcolm Baldrige Award.

Frana: Okay.

Hedger: MDQ. Market Driven Quality. That’s what that is. I was paralyzed there.

Frana: Right. I was looking at your class syllabus, I think.

Hedger: Right. Ah, MDQ, MDQ is market driven quality. It was IBM’s flavor of a quality program in the late ‘80s early ‘90s. Driven off of, it’s a “TQM thing” if you go back and look at the history of TQM. And it was driven off IBM saying we are going to get better at quality and it was actually driven off of the Motorola Six Sigma Programs.

Frana: Oh, okay.

Hedger: Okay. So it was a rather concerted effort to get across the board in development, to develop better products, a better quality of products.

Frana: And did that affect you on a personal level where there was a change in the way you did things because of MDQ?

Hedger: Not per se, except we reported our results regularly. And because of our bowers? winning we were off the IBM location that we would be compared to. You know we'll find out how those guys do it.

Frana: So you paid more attention then to process?

Hedger: And measurements and metrics, yes, yes and data.

Frana: To ensure that the end product would be of quality.

Hedger: And the goals that were set. Like in the MDQ program there were four or five goals. Reduce **ecars???** that's IBM external customer problems, improve customer satisfaction, no defective fixes, and reduce problem calls. And that was pretty much the same across IBM software.

Frana: Okay. All right.

Hedger: And we would measure that and report it.

Frana: Right. And that was around at some level before, but it becomes sort of, it was institutionalized in a way.

Hedger: Yes, it became an important program at IBM, quality programs come and go. They are in cycles. So this was the version of that.

Frana: I got you. I got you. They don't call it MDQ anymore. Do they call it something else?

Hedger: They call it something else. I am virtually certain.

Frana: Now I know you've lectured on campus here about software testing and verification validation, and I know you've quoted Edgar Dykstra's famous aphorism, 'Programming testing can be used to show the presence of bugs but never their absence.' How does this reassure the rest of us who are using this software or relying on it increasingly in our daily lives and our livelihoods for safety transactions, you know, that software is secure?

Hedger: Well, it's a good question. Current software process models go back to doing things, trying to find defects much earlier in the development process. Like make sure you have good requirements, make sure you have properly skilled and trained people. There's something called software inspections, which is a fairly formal way of reviewing code and various design documents and artifacts that get produced by the process, to look for defects much earlier in the cycle. So you don't have to, it costs about a dollar to find them in an inspection, it costs about ten bucks to find them in a test, and it costs about one hundred bucks to find them in the field. So we're trying to divide that up. Now that's a very formal process. It takes very disciplined organizations to make that work. So that's one of the things we are trying to do when you talk about testing isn't going to find them all. Because you are going to ship software with some problems, you're just not going to find none. Now maybe the customer won't find them all, though there's still some there.

Frana: Now when you were working on these various systems did you have periods when you actually froze the code? Like I know that I had several friends working on the Y2K problem and they froze the system and no one could actually add or subtract from it.

Hedger: A clear milestone in the development was to freeze the code as it went into its last test cycles. You really wanted to manage change whenever it goes in to test. So that what you are testing underneath is (stable?), and the software that you are testing isn't changing underneath you. Right. And then when you really want to freeze it make it very hard to change, is you could down towards the end, your ship date and your test cycles get much more, make it much more difficult to change.

Frana: And did that apply to the development cycle too? Were there times when people could, was it bundled, when you could add or subtract from the system?

Hedger: Yes, but you had to have a reason.

Frana: Okay.

Hedger: You could add code, but you had to have what we called, …Software changes. In the ‘70s, there was kind of an early period where you could check in, check out, and it was fairly loose, and then as you got to integration dates and made it available to other parties and other people on your team and software groups, you would have to have a bug report, a PTR, Problem Trouble Report, to integrate code, to add code to the system.

Frana: I see.

Hedger: So you were fixing a problem.

Frana: It was all documented and everybody was…?

Hedger: There was a reason why you were doing it. Right.

Frana: And part of that is, if you need the record because it could actually change, it could create another problem?

Hedger: Yes. You want to know why it changes and what was changed. And you also want to know if somebody reported the problem, like if a tester reports the problem, you use these PTR’s to find, to report the problem to the right people that can fix the problem.

And then the routing back. This started out as a paper process, today they are all automated.

Frana: It's all done.

Hedger: And you can buy these tools on the market.

Frana: Okay. Did you have direct contact with the customers, the clients that were actually using the end product? When they reported, you know, problems with the system, did you hear about that? Or at that point, was it out of your hands? And how much direct interaction was there between you and the end?

Hedger: Well, there were several levels of service or support between the customer and us. So they would use it, problems would usually come through service and severities? would be attached and a regular process for fixing problems, resolving those problems, was also addressed. So you didn't usually hear very directly from customers. The service was supposed to buffer development from them, from that work.

Frana: And can you attach a timeline to this? When you release the system at first there is this flurry of activity and then it grows quiet? Or does that never happen? Is there always a flurry of activity with problems being reported?

Hedger: At the initial release of a system, it depends on how many customers get on early. So it’s customer usage driven. But you often would get a spike and then problems reported from the customer would tail off over time.

Frana: Okay.

Hedger: So it would be very hard to say release two of a system is better quality-wise than release one with software.

Frana: I gotcha.

Hedger: Whereas hardware always has that goal. At IBM there is a corporate policy, the next release has got to be demonstrably better than the previous release of the hardware.

Frana: How do they measure?

Hedger: Reliability measurements and all the reliability stuff that goes on in building hardware.

Frana: Okay, And I take it, I didn’t ask this question because it seemed obvious, but there’s a lot, we’re talking a lot of software reuse between these various systems or are they all built from the ground up and all tested over again every time something is wrong?

Hedger: When you say various systems?

Frana: Well like, from System/3 to 38?

Hedger: There was very little, System/38 there were multiple releases, meaning probably once a year for its life, so there were probably several releases. That was very independent. 38 was all brand new. 32, 34, 36, I believe it was pretty much a stand-alone development effort from the System/3, though probably lots of similarities relative to the basic architecture.

Frana: Okay. Because they do talk a lot about software reuse now.

Hedger: Software reuse yes, comes back in, and you've got to design for it. Though the way we got use on the 38 was through something called "common macros and includes."

Frana: What are common macros and includes?

Hedger: Well, macros is a software thing to define a little chunk of code that others can use.

Frana: That's a macro.

Hedger: That’s a macro. And they go way back. They go way back somewhere in the early ‘60s. So macros was a way, so if I can, this little routine, if we take the square root it would be a macro. And the ‘include’ was just another chunk of code, I want to reuse this code. So if you design for them, and made people aware of them, there was a significant amount of kind of detailed reuses of kind of standard pieces that had been agreed to by design to use these pieces. So it was an early shot at reuse.

Frana: So you had a macro or a routine library?

Hedger: Yes.

Frana: And was that right there available to you on site, or you had a network connection and you’d get it from some sort of central ..?

Hedger: It was in the software libraries, that when you went to compile a module, you would call a macro; it would go find those macros.

Frana: Okay. All right. That’s great. This may be a segue here. Watts Humphrey starts his book, *Managing the Software* Process—have you read that book?

Hedger: Yes.

Frana: Back in ’89. Watts Humphrey is a supporter of the Charles Babbage Institute, that’s why I know his work. He starts with two proverbs: ‘If you don’t know where you are going, any road will do.’ And: ‘If you don’t know where you are, a map won’t help.’ What do you think his book and his software engineering style or software management style I guess? Do you have personal proverbs that you’ve developed?

Hedger: “Inspect what you expect.”

Frana: “Inspect what you expect.” Okay. And is there a story related to the development of that personal motto?

Hedger: No, I just learned you will get surprised if you don’t go off, if you’re expecting something to happen if you don’t go check on it to make sure it is happening you will be surprised. So you continually inspect your expectations. Or another way of saying it is if you’ve got somebody off, a group off working on a software project and you don’t pay attention to what they are doing, don’t go ask once in a while to check on what they are doing, you will be surprised. [laughter]

Frana: Have you ever met Watts?

Hedger: Yes I have. Oh yes.

Frana: Yes, and what was the occasion?

Hedger: Well, Watts was my peer when I was out east. I was out east for three years, 1982 to 1985. In my last year, I was the manager of a group software staff function that managed quality process and tools for all non-370 software development. Watts had a similar group; only he had all the 370, the large systems. So he had **MBS ?,** and **BM?** , and **BSE?** Operating systems reported, or worked? for him. So he had sixty people and I had three.

Frana: Okay. Now did the two of you interact quite a bit?

Hedger: Yes, we did. Yes, quite a bit.

Frana: Did you have different management styles? Or similar styles?

Hedger: Watts was a very senior executive at IBM. He had been an early manager of 360 and had lots of experience. And he successfully took all he was doing in Poughkeepsie for software process improvement, and when he moved to the Software Engineering Institute at CMU,…

Frana: Right. He's not there anymore I guess.

Hedger: No he's still there.

Frana: Oh is he?

Hedger: Oh yes. He’s still there. He brought that with him. Essentially, the capability and maturity model of SEI is essentially what he was doing at Poughkeepsie in the early 1980s. He was assessing IBM software sites that same way.

Frana: All right.

Hedger: Yes, I saw Watts at New Orleans in March.

Frana: Oh you did?

Hedger: Yes.

Frana: His Web page is missing in action. I was looking for it. Someone wanted to find it and I couldn’t find it. Has he moved somewhere now, within the organization there?

Hedger: I don’t know. He’s a fellow now so he’s kind of, but he’s off on some …

Frana: SEI?

Hedger: Software Engineering Institute. And he’s off working from the ground up. So he’s trying to teach software engineers how to do software engineering at the grass roots.

So he’s got classes and disciplines I guess you’d call them, called “personal software process PSP” and “team software process TSP.”

Frana: And what do you think of those two ideas?

Hedger: They are great, but they need to be taught at undergraduate level in computer science schools. It’s very hard to get those in to an organization.

Frana: Unless you teach them fresh?

Hedger: Unless you start very early. But it is very good discipline.

Frana: And we haven’t been doing that today.

Hedger: No.

Frana: You teach sometimes with your colleague **Steven Kahn ?.**

Hedger: Yes. **Steve Kahn**.Yes.

Frana: And he’s divided, I read something, he’s divided the history of software engineering process in to several different eras?

Hedger: Yes.

Frana: He has the history by decades: the ‘60s he calls “functional”, the ‘70s are the “schedule era”, the ‘80s the “cost era”, and then the 1990s, I guess, the “quality era.” Is this a fair way of looking at that history? Do you have another way of sort of separating out the trends? Are there other important trends that we maybe want to look at or consider? I know you mentioned on the phone extensibility as something important.

Hedger: Well if you look at quality as more than just bugs, right, quality is reliability, usability, and maintainability. Anything it takes to make a good piece of software do what the customer needs to have done. So I would make quality a little broader. We are probably learning a lot more about that now with graphical user interfaces. So the design methodologies to produce software, there’s the ‘70s, ‘80s and ‘90s, technologies to do that have changed and the kinds of applications it can produce have changed. So we’ve gone from green screens, command based processing, to very sophisticated user interfaces. So that all kind of drives a little bit of this as well. But the needs for reliable, easy to use, solve the customer problems software, that essentially the system could be stuck away in a closet and run the customers’ business, is still there.

Frana: All right. And what is it about object-oriented programming that changes the sort of functionality of these systems?

Hedger: Object oriented is just another way to do design.

Frana: Okay. Does that have a lot to do with the graphical user interface? I mean are those all considered to be the objects?

Hedger: Pieces of them can be considered to be the objects. Yes.

Frana: Is that a radically different way of looking at programming or do you see it as sort of an evolutionary process?

Hedger: When you make the switch from structural procedures or structural design practices, and make the design, the switch to object oriented is a fairly significant switch in learning some new techniques to do things.

Frana: Okay. I know the kids learn OOP from the start, but you probably had to learn this on the job.

Hedger: Yes, we learned it with, we started with flow charts, I think it was.
[laughter] So there have been lots of design practices that have evolved since then.

Frana: Well I learned flow charting too, is that still part of the curriculum? We always found it to be useless as students. Why can't we actually sit down and write the code right off the bat. What are we messing around with all this flow-charting for? Is that still a priority?

Hedger: Well, design is. Design is. But they all want to rush, it's not just students, everybody wants to rush to code.

Frana: But still you don't see that you cannot short-circuit the process.

Hedger: To build scaleable, extensible, reliable software that is going to be used by a large number of people? No, I don't think so.

Frana: Now you do some consulting, do you see people doing this? Trying to short-circuit this?

Hedger: Oh absolutely.

Frana: Yes? All the time?

Hedger: All the time.

Frana: That usually doesn't work? Or it only works for the limited purposes for which it is intended.

Hedger: Yes. And the thing to remember is that if software is going to be successful, a large part of the cost is in maintaining it and in keeping it running and adding enhancements to it. And the more productive you can make that piece, the more successful your software is going to be.

Frana: What about the other big innovation, distributed processing? Does that change the way programming efforts are made?

Hedger: It certainly changes the environment in which a software engineer does program.

Frana: Did you ever work on client-server kinds of applications?

Hedger: Some of the early stuff. Hooking PC's to AS 400s.

Frana: And some of that was obviously done.

Hedger: Yes.

Frana: Another question that came up at the software history conference last fall: Someone asked if embedded software, embedded systems is rebundling. Is there such a thing, is unbundling a real thing, or is it just sort of a nice way of, or a way of pleasing the Justice Department? Is it really possible to think of hardware and software as separate things, as separate entities?

Hedger: Yes. Absolutely. They are separate products. And remember applications in the bundled days often used to be given away. IBM had Class 3 programs in [their] program library [and] essentially they gave them away. Customers could have them.

Frana: Just like the manuals? Anyone could have them?

Hedger: Yes, right. Now depending on where you break the machine operating system interface, there are various reasons for doing that in computer systems, but often in embedded systems you are doing a particular chip with a particular function to make a device work.

Frana: It's very special purpose.

Hedger: It's very special purpose, right. So, it would be hard, it's making a device work, again making a product work, like a TV or a toaster.

Frana: But microcoding is kind of hazy territory isn't it? You don't think that's, I mean it gets hard wired sometimes.

Hedger: That's just how it gets shipped.

Frana: Okay. So that's the end product? But the process…

Hedger: But the process can be almost absolutely identical except when you come to testing it. Testing it may be different because your testing devices, what you’ve got to test it on is different. But the process to design, to get requirements to design, implement microcode and the supporting processes of configuration management and problem tracking and all that sort of stuff, doesn’t have to be any different. The difference will come sometime in how you test it. And in fact, you may simulate it before you get the hardware.

Frana: Before you even get the hardware?

Hedger: Because there is , depending on what you are doing and what kind of microprocesses you are running on, you can get simulators ??

Frana: And it is easier to test the simulator than to test…

Hedger: Oh yes, probably, because you have to have a way to …yes, right.

Frana: Are there big picture things that I have missed here? Details of your career that you really want to communicate in this oral history, this interview…things that occurred to you just in this conversation that are important and we haven’t talked about yet?

Hedger: I think better completely to understand the System/38 development and its architecture thing, like I mentioned at supper, Dr. Frank Soltis, on the hardware architecture side would be a good guy to speak to.

Frana: Do you know where he is now?

Hedger: He's in Rochester.

Frana: Oh, he's in Rochester.

Hedger: Yes.

Frana: He's your hardware compatriot?

Hedger: Yes, he's kind of on the hardware side. So I think he would be one person that should be added to the list. There are other names that were early on that Glenn could probably fill, if we get to Glenn Henry he could probably fill some of those names in. Because this organization evolved over time. **???** done.

Frana: Who was the general manager when you were down there?

Hedger: General manager?

Frana: Or who was the administrator of the site?

Hedger: Well, there was a lab director. The lab director ran the lab and the lab director was Harry Tashijan, and before Harry was a guy named Hal Martin.

Frana: Hal Martin. Okay. No idea where they ended up?

Hedger: Hal Martin died. And Harry Tashijan, last I heard was in Princeton, New Jersey. But he was kind of the father of this whole line of systems. He was an engineer, but he was kind of the system manager. In fact he was a super-system manager. At one time there were three systems managers for this family of products and they all reported to him and he reported to the lab director. Back somewhere in the middle ‘70s.

Frana: So he would have been your, which line would he have been to you?

Hedger: Well I was a middle manager so, second, third, he was a fifth liner, by gosh. He was unique though.

Frana: And there’s never been anyone that really had that kind of responsibility since?

Hedger: Well he’d be a lab director.

Frana: Right.

Hedger: Yes. Hal Martin later became site general manager and he passed away 1978 or 1979 on a vacation in Mexico, Christmas time.

Frana: Unexpectedly?

Hedger: Yes.

Frana: Well, thanks Dick.