TALKING POINTS

An Interview with Jerry Weinberg

BETH LAYMAN
associate editor

What should a person say when introducing Jerry Weinberg? Should one say that he is an icon, a leader in the field of software engineering? Brilliant? The author of more than 30 books, including *The Psychology of Computer Programming*; *Handbook of Walkthroughs, Inspections, and Technical Reviews*; and *Quality Software Management* (four volumes)? Or should one say that he's just a really nice guy? He is all of these things and more to many of us in the field. His workshops and writings have taught, encouraged, and inspired us over the years. I was thrilled when he agreed to let me interview him for *SQP*. We talked about the state of the practice, becoming a technical leader (incidentally, yet another title of one of his books), and the future of software engineering. I hope *SQP* readers will find his answers as insightful and entertaining as I did.

Weinberg on Software Engineering State of the Practice

SQP: **What do you consider the major milestones of software engineering discipline in the last three decades? In other words, what concepts or methods introduced and adopted in the last 30 years have changed the face of software engineering most dramatically?**

JW: Well, I don't think there have been any.

SQP: **Really?**

JW: Yes. Well, I think, first of all, if you really mean the "adopted" part, in the sense of changing the whole discipline, we haven't reached that stage yet. If you review the information from surveys like the Software Engineering Institute (SEI) conducts, by far, most software organizations are operating at what they call level 1 and what I call level 0. Sure, we hear and read about companies that are doing good things so there are concepts that have been adopted—things that would change the industry if they were adopted by some substantial portion of the people, but right now our industry as a whole has not changed at all.

SQP: **What about things like reviews and testing—surely these are more widely practiced?**

JW: Well, certainly, testing is more recognized as a separate function now. And, that has helped certain organizations. But, in many organizations, it has just made them sloppier developers; they are just more encouraged to throw stuff over the wall to testing. So, I would say that there may be some new testing concepts here and there, but nothing that has swept the field even 50 percent or 30 percent.

SQP: **You've written a number of books advocating various concepts and methods. Do you believe there are any that, if adopted, would change the face of the state of the practice?**

JW: Ah, now that's a different question. I'm reminded of this farmer that one of my former students, who is now an ag [agricultural] extension agent, told me about. My student was out introducing some new technology for something like planting corn. He invited this farmer to a meeting to discuss the new technology, but the farmer said, "No, I don't think I'll go." My student said, " But, why not? This is something new that would improve your crops!" And the farmer said, "Oh, I already don't use half of the things that I know would improve my crops, so what's the point in getting another one?"

I'd say that's about the state of where we are. We have a few things, a few fundamental things that if people would do consistently, then we could start to take off. One is requirements work. Now, I'm not just talking about sitting down up front and writing down everything you want. There are clearly cases where this will not work. Barry Boehm has written some good things about the spiral model and how you use risks to decide how deeply you go into requirements.

So, requirements work, work that we know how to do, work that some people are doing very well, is one, and reviews is another. Without reviews, requirements work doesn't mean anything. Third is configuration management. And, we've made some progress in these areas. We know more about them, we have people doing things better in all these areas, but it hasn't swept the industry.

You get a wrong impression if you read the journals. You may conclude that, because this one group is doing it, everyone is. And that's just not true. I think that, over time, there is some natural selection process that will take place, but that takes time.

Another thing that's happening is this: If the growth of the software industry slows down a bit, then we have a chance for people to become more experienced managers, and they will implement these things. Up until now, things have been growing so fast that there are simply not enough people with enough experience to bring the wisdom that we know to bear in very many places.

You know, someone goes to college and takes a computer science class. They learn Perl scripts or Java and they get a job making $100,000 a year. "Chief architect!" A nice title, but what's in a name? They just don't have enough experience with the kinds of things we are trying to do to keep our profession growing more successful.

You know the Peter Principle, I'm sure, where you rise to your level of incompetence. But do you know about the Paul Principle?

SQP: No, what's that?

JW: Paul Armer, who was one of the early leaders in the software business, said it's not just that people rise to their level of incompetence. In a situation where the problem space keeps growing and people stay where they are, then the level of competence that is needed rises past them.

That's more a description of the situation we have today. I'm not saying we were better 30 years ago. I'm saying that, relative to what we had to do, we were probably just about as good. But I think that people today, if they went back with the equipment and tools and knowledge they have today and just solved the problems we worked 30 years ago, they would probably do much better. But that's not good enough. There's a crisis of rising expectations going on.

SQP: What are your feelings about the use (and misuse) of software engineering standards such as those from IEEE and quality/process models such as the Software Capability Maturity Model (CMM)?

JW: You can add into that other things like eXtreme Programming or anything that has really good ideas behind it. We know that there is no tool that anyone has ever made that someone didn't manage to misuse (and actually make a business out of misusing). If we ruled out things because of their potential misuse, we wouldn't be doing anything! We wouldn't have hammers, knives, and forks. So, you've got to be careful. You've got a lot of people writing a lot of stuff about how a technology is misused, and you might think that because the technology can be misused, we shouldn't pay any attention to it. But I think that's the wrong orientation. I think you can write about how something can be misused as a warning to people who are going to try to use it. However, I think that we need to concentrate on how to use the things well, if we are going to use them at all, and for the right reasons.

Take the CMM, for example. I think a lot of misuse starts from a misapprehension on the part of management at some high level. For example, if you're told that if you are not assessed at a certain level, you can't get contracts from us. Well, that really wasn't the point of all this. As soon as you couple certification with assessment, you start getting people cheating on the assessments. And, that's happening all over the place. Same way with standards—standards are a wonderful concept. But, if they are used to make people conform as a marketing device (our standard imposed on everyone else), then they backfire on you.

Then, with things like eXtreme Programming, when it gets built up like it's a magic solution to the problems we've been having for years and years (problems caused by not doing things like requirements, reviews, configuration management) then it's not going to help, it's only going to make things worse—this pursuit of magic. And that's what we see over and over again.

I think that all three of those things (CMM, standards, eXtreme Programming) have great promise, and I've seen them used well, but we're not using them enough—and for the right reasons.

SQP: Any thoughts about how to increase their use?

JW: My own conclusion a long time ago was that their use will increase when we have leaders in the field whose influence is aligned with their knowledge. Then we wouldn't have the case where people who really don't understand, but are very persuasive, are persuading people to do the wrong things for the wrong reasons. Or where the people who really do understand are very poor at negotiating, presenting ideas to people, and persuading people. Today, we're driven by who is the best persuader, not who knows best what to do. So, we have a real crisis of leadership, which we've always had because we've been growing so fast.

SQP: Based on your decades-long observation of the field, how would you compare our rate of progress in the technology to our progress in the sociology of software development? Do managers better understand the human and team aspects of software development today than 30 years ago?

JW: Interesting question. I think that, more often, you'll get them saying the right words than maybe they did 30 years ago! In practice, I think we saw much better teamwork back then, although managers didn't know to call it that or know what they were doing to inculcate it. Today, there's more lip service done to teamwork, communication, and so on, and they all nod their heads. But, I don't think the average manager of software development today really knows more about that in the practical sense of what to do.

That doesn't mean there aren't some outstanding people; we have many more outstanding people now than then. But, we also have many more times the number of people in the business.

WEINBERG ON PROFESSIONAL DEVELOPMENT OF SOFTWARE PROFESSIONALS

SQP: If you had to guide a young professional to prepare for a successful career building software, what balance of training and experience would you recommend (for example, mathematics, computer science, engineering, mountain climbing, team sports, and so on)?

JW: It's not a matter of what the subject matter is, it's a matter of how they approach the subject matter. I think we need less theory and more practice. I think a lot of the engineering schools in other disciplines have come to understand this. I have a number of friends whose sons and daughters are in wonderful engineering co-op programs, where they spend a little more time in school in elapsed time, but every six months or so, they go off and work in an engineering firm. They have something real to tie things to. For example, over the years, we've done our problem-solving leadership classes a few times with university students and it doesn't go well with them. Yet, it's outstanding with people who've worked in the field. It's because they have no real experience to tie stuff to whether they're social concepts or technical concepts. These students just come out of school with way too much theory of one thing and another and not enough sense of how these would really work and what it would take.

Those experiential programs are very, very helpful, in my opinion. But, there is such a hurry to get ahead. Nobody wants to go back to school. And, companies don't want to invest in that, even for a week's worth of training. I get people telling me, "Oh well, there's no point in paying for training our people, because they just leave." Well, of course, they leave because they don't get training.

I would also say to a young person—and this is real for me because one of my sons has had a career in this business and I've always advised him this way—only work for good organizations. If they're not well managed, then get out and go to a better place and be around the best people you can. One time he was working for a company that did government contracts and they asked him to falsify the hours he worked so they could bill against a certain contract. He called me up because he'd never heard of anything like this and I said "just leave" and he did. You spend too much time in a place like that and it just spoils you from ever doing really good work. That kind of lack of integrity isn't necessarily widespread, but there is enough of it around that you don't want to get yourself caught in that.

Similarly, with incompetence, in management or in technical things, my advice is always work with people who are better than you and then you'll grow. And, if you find yourself not growing, you should move on to something else, go back to school, or go work for somebody else. This is how you develop and this is how the field develops.

SQP: That's such good advice. I feel lucky that I've been able to do that in my own career.

JW: Yeah. Me too. Me too. And I have violated it enough times, not so many but enough times, that I know it's practical advice and dangerous advice to ignore.

SQP: How about advice for growing new technical leaders or advice on becoming a technical leader?

JW: OK. The first thing I would advise, is, if you don't want to do that, don't do it. One of the problems we have is that people are not even very far along in their technical careers and then their managers say, "Well, we need a manager so we are appointing you; you're our best technical person." And, they don't really want to do that. As you get older, you might develop a sense of "Well, I'd like to be more involved in leadership." Some people might have this sense when they are young, but it typically goes in more traditional fields that there are older people around to play these roles. But here, they push people too fast. If you don't really want to do it, don't do it. It doesn't mean you won't want to do it later.

If you're not sure, take the opportunities to practice without making an official step. For example, if you're involved in the bowling team, maybe you could become the captain of the bowling team. Or you might work with Boy Scouts or Girl Scouts as an adult leader, or work for some charity. Or take out a short-term task force assignment at work where you get to practice a little and see what you like and don't like.

So that's the first thing—if you don't want to do it, don't do it. The second is if you're not sure, get yourself into situations where you can find out and back out. And, the third thing is to have some children! I mean actually raise them—I don't mean just go and propagate the species!

We've been finding over and over again that women in the work force who raise their kids and come back to work as project managers (women in some hotshot software companies) were very successful. So I interviewed a number of them and one of them put it very well. She said, "Well, you know what, running these projects is just managing a bunch of teenage boys. I've already done that. I know the games they play and I know how to handle it." It's true. Not everybody has to have kids, but having kids is a way to learn about dealing with people and being responsible. You could get this in other ways. You could have dogs, for instance. My wife is a world-renowned animal trainer, and she teaches a course called "Dogs and Their Managers." She comes in with the assumption—and she'll be doing some of this at the AYE conference—that if people dealt with their people at least as well as they did with their dogs, they would already be better managers!

SQP: Speaking of AYE, why are you leading the effort to hold the AYE Conference? What effect do you see it having on the software industry?

JW: It's exactly addressing what I was talking about. It doesn't just look at technical things with a side comment or two about "Oh yeah, and this is how to implement this." The name of the conference is Amplifying Your Effectiveness (AYE). It really concentrates on personal effectiveness in a technical context.

So, if we go back to some of the things we've been talking about, if you were introducing the CMM for example, the sessions would be about what you should and should not do, what you need to learn, and what you need to practice in order to be effective at that. And, it's also very experiential because of that. A bunch of us were having a discussion about how inadequate most conferences are; people stand up and read this little technical idea and that little technical idea, they're all very neat, and you think "how clever." But, they are not really making a difference. So, we said, "Well, what would make a difference?" We said, "If people were more empowered with specific skills for how to make things happen, how to nurture good things, how to let the things that are not sensible fade away, how to handle the one person who seems to be disrupting your whole effort to improve software development." This year, we have several sessions on coaching; how to run effective meetings of various kinds, like review meetings, and how to negotiate.

Additionally, there will be lots of role-playing and simulations, as with all our courses (very little sitting and listening). And, as with last year's conference, there will be sharing of experiences, too. People came with problems, for example: "I was trying to get this done and it didn't work and I don't know why." Other people will comment on what they have done and advise. There will be some models presented, but mostly models of social architecture. I'm doing a session with Bob King on "getting a project started right." It's about stuff that people do before they even realize they're starting a project that gets them off on the wrong or on the right foot. And we will do something that let's them pretend they are starting a project and notice what they do and what it leads to. We have another session on life cycles. But instead of giving the theory of life cycles, we're going to actually act out various life cycles and trace not just the product life cycle, but the life cycle of errors in a product; where they start, with people playing the role of errors and seeing where they are introduced and where they are found. And also the life cycle of the people doing the work. How does it affect their careers, and how does it affect what they learn and how they improve? So, we'll actually be walking through those together.

SQP: **Sounds entertaining and different.**

JW: It is. People have lots of fun, but they also come back with things they can really use.

WEINBERG ON FUTURE DIRECTIONS IN SOFTWARE ENGINEERING

SQP: **It seems that nonprofessionals are creating more and more software and systems, and that "programming" is being accomplished at higher levels of abstraction (for example, spreadsheets, GUI generators, wizards, and so on). What are your views on how the expansion of software development to an "every man" activity affects software quality results and software quality practice.**

JW: Of course, you mean "every woman," too?

SQP: **Of course!**

JW: I guess my answer is, "It's the best of times and the worst of times," to quote Charles Dickens.

The best of times is because, well, think about the restaurant business. I used to travel around with my father when I was a kid, and he introduced me to the concept that certain towns were good restaurant towns and others were not (of course, this was before chain restaurants; I'm not sure it's that way now). But, what the determining factor was, in certain towns, the people who lived there were not very discriminating diners. And, where you didn't have discriminating diners, you didn't have good restaurants. There was just nobody there to demand better food. An example from today is, you have hotels at big airports and the hotels have a restaurant. Now, the only people who stay at these hotels are the people whose flights have been cancelled. They are stuck there, they don't have a car, there's no place to go. So, they eat in the hotel restaurant and they are never very good because the customers don't have a choice, they have no way to express dissatisfaction, they would never come back anyway, and if they don't eat there, they don't eat.

That's similar to the way it was 30 years ago. If you didn't like what you were getting from your computer/IT department, that was just too bad. It's like you just didn't get to eat. And now, there are lots more choices including, "Well, I can just put a spreadsheet together myself and it may not be great, but I'm in control of it." Whoever proposes to do stuff for this person on the outside better be substantially better than that. Otherwise, "I'm gonna cook at home and not go out to eat." Over time, this works to good effect.

The second good thing about it is this: People who might not have ever thought of going into a career in software development have more chance to find out about it. When I was a kid, and this was more than 30 years ago, I had read a couple things about computers and I knew that I wanted to work with computers. So, when I got to college I went to the guidance counselor and said, "I want to work with computers." Well, they had not even heard of computers let alone have a computer science curriculum or people on campus who had seen one or used one.

Nowadays, more people will have an idea of what it's all about and maybe we'll get more people coming into the field who will be good at it. Before, we only got people who were math hotshots and there's a whole lot more to the business than mathematics.

The worst of times is the same thing of course. People will put together a spreadsheet with one or two columns and formulas in it (column A plus column B) and now they think they are programmers. Then, if you're building embedded software in a pacemaker they think, "Well, that's just the same thing." They may have unreasonable expectations and impatience with the difficulty of really getting quality results. So, that's the good and the bad of it.

Before there was spreadsheet software, I actually wrote a number of spreadsheet programs and that was fairly difficult. I wrote them in assembly language and I can tell you that it's just not productive. It takes a certain amount of brains and capability to do that and which should not be required. Professional software people really shouldn't have to be doing that.

As another example, we have an energy crisis now. Imagine if we made the people who know how to run electrical power stations go out and fix wiring or change people's light bulbs! Then they wouldn't have time to attack the really big problems that only they are qualified to solve. So the fact that people can now change their own light bulbs is a good thing for the profession!

SQP: But, heaven help us when the spreadsheet developer builds software for a pacemaker!

JW: Yeah right, then you're in big trouble. There is a story they told me at Microsoft. Some years ago they found out that there was this hospital that was running their intensive care unit off an Excel spreadsheet! And, they were horrified, because it wasn't built to be life-critical software. So there will be things like that and that's the downside.

To learn more about the Amplifying Your Effectiveness (AYE) conference, visit the conference Web site at www.ayeconference.com. To learn more about Jerry Weinberg, visit www.geraldmweinberg.com.

How helpful did you find this article? Please provide feedback at our online reader survey, which will be active for approximately ten weeks after this issue's publication: www.asq.org/mr/sqp3_issue4.html .

© 2001 American Society for Quality. Reprinted with permission.