



Presents
An IT Metrics and Productivity Journal Special Edition

Focus on Dr. Stephen Kan, IBM
A CAI State of the Practice Interview
August, 2005

Biography of Dr. Stephen Kan

Dr. Stephen H. Kan is a Senior Technical Staff Member (STSM) and a technical manager in programming at IBM in Rochester, Minnesota. He is responsible for the Quality Management Process in software development for IBM's eServer iSeries covering all aspects of quality ranging from quality goal setting, supplier quality requirements, quality plans, in-process metrics and quality assessments, to reliability projections, field quality tracking, and customer satisfaction. Dr. Kan is the author of the book *Metrics and Models in Software Quality Engineering*, numerous technical reports, and articles and chapters in professional journals. He has participated in and led many software project and process assessments over the past 15 years. He established the CMM strategy for the iSeries software organization and led the process improvement effort with a core team in achieving CMM Level 5 assessment in 2004. Dr. Kan has been a faculty member of the University of Minnesota Master of Science in Software Engineering (MSSE) program since 1998.

◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆

CAI: Could you tell us a little bit about yourself, your early career, and how you got to be where you are today?

KAN: Before starting my career with IBM, I had already had several positions as a statistician and a programmer and had worked for almost 10 years in the industry. When I started at IBM, I was hired as a software developer but when people found out about my background in statistics, they moved me into the area of quality and process improvement. Currently, I am a Senior Technical Staff Member (STSM) and Technical Manager for IBM's eServer iSeries software quality. I also lead the software quality process improvement effort across IBM's Systems and Technology Group (STG).

CAI: You've written a book- *Metrics and Models in Software Quality Engineering*- that is considered the single best book on software quality engineering ever written. What did you set out to achieve when you began writing this book and what were some of your influences leading up to it?

KAN: Prior to writing this book, which was first published in 1994 (1st edition) and then again in 2002 (2nd edition), I was publishing papers in IBM's Systems Journal and also presenting at various conferences. It was an editor at Addison Wesley, who had been attending these conferences and reading these papers of mine, who invited me to write the book. My original intent was to write about metrics and models from a quality practitioner's perspective and to keep the context as experience based as possible. That was really my key objective- to focus on experience based metrics usage and to use a lot of real world examples.

CAI: What advice do you have for organizations that are new to software metrics? What are the critical success factors? Are there any specific metrics that you would recommend starting out with? How would your advice differ for small organizations as opposed to large organizations?

KAN: To start a new metrics program you must start off small. You've also got to be project based; that is, you must begin with specific projects as opposed to taking a broad-based top-down approach for the entire organization. You must start out small, with only a few metrics that are effective, and you must get early successes in place.

Metrics for effort, size, schedule, and cost are the basics. Additionally, metrics to measure quality are most important and should be among the starting set. I want to emphasize that the quality and effectiveness of metrics are much more important than the quantity of metrics you decide to track.

Furthermore, for any software metrics program to be successful the approach must be analysis driven. That means that the full spectrum must be traversed, from raw data collection to analysis to the formulation of actionable data to process improvement conclusions. We need to recognize that there are metrics programs out there that just collect raw data and that maintain only a shallow focus on analysis. These tend not to be very effective metrics programs. A truly analysis-based approach, in contrast, will derive meaningful information and actionable knowledge from the metrics and data. You've also got to make sure you have good people with solid training in data analysis and measurement. And you've got to have strong executive management support.

A key caveat for small organizations would be to avoid the use of control charts, at least until there is some experience established. It is my opinion that even large organizations, when they start out with a software metrics program, should stay away from control charts. Control charts are being discussed a lot these days in relation to the CMMI but in my view, people should not be using control charts until they have accumulated a very rich repository of historical data and understand what the data mean.

I would also take a much more visual, more graphics-based approach when presenting your metrics (as opposed to using tables of numbers). The visual impact will be greater.

CAI: For those IT organizations that have effectively integrated processes and metrics into their software operations, what would be your characterization- in aggregate- of the resulting improvements, in terms of ROI?

KAN: ROI has been very high, anywhere from 10-1 to 100-1. By way of example, if you had a software organization or an IT organization with 100 people and you were to employ what we call a "few good persons approach," which involves one or two people driving the metrics program, your returns could be enormous.

There seems to be a misperception about this in the industry. People seem to think that you can't realize these kinds of metrics and process based returns until you have already reached a very high level of maturity. In reality, however, the best time to start a metrics program is as early as possible.

CAI: If you had to take a domestic IT organization at CMM Level 1 and recommend a generic plan of action for improvement, what would be the first four or five things that you would have them do immediately? What would you realistically expect to get out of your effort and how would you define that value in business terms?

KAN: I would recommend, first of all, a very strong focus on project management. From having read so much of the literature in this industry, and from 15 years of personal experience conducting project assessments on many, many projects, I can tell you that the major reason for project failure is poor project management. Consequently, we must train our project managers to become professionals. Much more importantly, we must train them to manage quality. I still see a lot of project managers who are just trying to manage the schedule of a project as opposed to the quality.

Second, I would deploy a metrics program with only a few metrics and I would make sure that these metrics are useful at the project team level. Then I would start using these metrics to track project progress and to conduct analysis.

Finally, I would focus on improving three things: the development environments, the tools and the support infrastructure. The objective here would be to establish and improve the organization's standard processes, as based upon actual project experiences.

CAI: It is frequently cited that 80% of IT spending is directed towards the running and maintenance of existing systems and infrastructure, as opposed to new development. Despite this, we still see all of the best thinking and publishing in our field- about metrics, about estimation, and about processes- being done in the area of new development as opposed to maintenance. Why

is that the case given that so much more of the money is being spent on maintenance? Do you agree that this is an area of opportunity?

KAN: I definitely agree that this is a huge area of opportunity. As for what is at the root of the discrepancy, there are certainly a whole host of reasons but I will point out just a few.

First, new technology and new development will always get more attention than software maintenance. That's just human nature. Second, developers, designers, and architects will always have much better career paths than testers and supporters. So career mobility comes into play as well.

These are not intractable problems, though. At IBM, management recognizes and understands these issues. Consequently, they put a great deal of importance on various other roles and disciplines within the software engineering community. As a result, there are a lot of testers here that have moved up to become distinguished engineers.

CAI: From a process and metrics perspective, how would you address the software maintenance issue? What questions should we be asking ourselves in order to re-apply development best practices over to the world of software maintenance? What specific metrics would you advise organizations to track if they were focusing primarily on maintenance as opposed to new development?

KAN: From an overall process perspective we need to deal with the career mobility question. We need to make sure that people working in maintenance have the same career opportunities as those working in new development.

In many IT or software organizations you will have one team doing development and one team doing maintenance. However, you will sometimes see organizations in which these two groups have been linked together and right at the component team level, too. So within the very same component you will have some people doing development and others maintenance and then everybody gets moved around. There are usually mutual benefits associated with this approach because your workforce acquires complete knowledge from both perspectives. In my observation, doing maintenance this way gives you much higher level of quality and productivity.

As far as process improvement initiatives go, many of these methods are the same for maintenance as they are for development. Root cause analysis, evaluation of process effectiveness and efficiency, Plan-Do-Check-Act, benchmarking and best practices can all be equally well applied to maintenance as to new development.

And when it comes to metrics, there are some very important metrics that are specific to maintenance. For example, what defect level are we at when we return fixes to the customer? When we fix customer problems, are our fixes defect free? This is

very important because when your customers report problems to you they are already in a state of displeasure. When you then "fix" the problem and it still turns out to be defective- this is highly detrimental to customer satisfaction. Another example is fix responsiveness. Were we able to fix the problem in a timely manner? Does the defect cause a system outage? How much system down time are our customers having? Do we deliver the fixes on time per our commitment to our customers? What is the defect backlog? By using metrics like these you will improve your software maintenance process while at the same time linking that process to overall customer satisfaction management.

CAI: So much is being made of offshore these days. What is the importance of metrics and process in an offshore relationship? What should organizations be doing to get this right? What should they be looking for?

KAN: The role of metrics and process is very important in offshore management, especially when it comes to tracking project progress and providing project status. Great distances necessitate a reliance on quantitative data and metrics.

I am by no means an expert on offshore, but my reading of it right now is that too many people are focused on the labor rates and on the cost savings aspect. I would take a much broader view. I would look at overall productivity, effectiveness, and efficiency. I would also look at quality. If your development team can't deliver acceptable quality, it's going to cost you a lot more in rework and maintenance and these will all be hidden costs.