



**Presents**  
**An IT Metrics and Productivity Journal Special Edition**

**Focus on Don Shafer, Author & CTO**  
**A CAI State of the Practice Interview**  
**March, 2006**

**Biography of Don Shafer**

Don Shafer is Editor in Chief of the IEEE Computer Society Press. He is also co-founder, corporate director and Chief Technology Officer of Athens Group, Inc. Prior to Athens Group, Shafer led groups developing and marketing hardware and software products for Motorola, AMD and Crystal Semiconductor. In the past six years he has led Athens engineers in the analysis, verification, validation and simulation of drilling, safety and positioning systems for deep water oil platforms. Shafer's work experience includes positions held at Boeing and Los Alamos National Laboratories. He is a Senior Member of the IEEE and an adjunct professor in graduate software engineering at Texas State University.

Our interview with Mr. Shafer was conducted in December of 2005.

◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆

**CAI: Could you tell us about yourself, how you got your career started, the path you took and what you're working on today?**

**DON SHAFER:** I've been in technology since I got my Bachelor's degree in 1970. Today I'm the Chief Technology Officer for a company called Athens Group. Athens Group is an all employee owned company in Austin, Texas. There's about 50 of us in there, 6 of us started the company in 1998, and we do a number of different software and strategy related types of engagements. I focused quite a bit on opening up our second office in Houston, and spent almost 30 months there. We focus on oil and gas in Houston. As it relates to software, those oil and gas guys are kind of where we were

in the 1970's.

My Bachelor's degree is in engineering from the Air Force Academy. I got out of high school in 1966 and went into a military academy to avoid the draft. It was a good way to get an education and not end up in a trench. Once I graduated from the Academy, I got my pilot training, had a very unrewarding relationship with a parachute, subsequently got out of pilot training, and then went into to the Air Force intelligence school.

I wound up spending two years building and managing the data collection network on the Ho Chi Minh trail. We had sensors on that trail, and we had IBM 360/65 computers, the biggest ones in the world at that time, and everything was all networked together at the Nakhon Phanom Air Force Base in Thailand. Everything was online and when night time came we would bring everything up and watch in real time all the activity on the Ho Chi Minh trail and this enabled us to call in air strikes. All of this was done with software and technology from the 1960's. And it was stuff that we're using today, the same way we do a lot of remote sensing, the same way we do data collection. The only difference is that we're using satellites today. At that time we weren't using satellites, we were using aircraft, orbiting aircraft. But it was basically the same idea. You had sensors on the ground transmitting data. And there was also a lot of database work, a lot of scrubbing of this data.

I got out of the Air Force in 1972, and went to work for Boeing on the AWACS project. That's when Bob Glass and I worked together. I also worked with Barry Boehm at that time, who was at Aerojet General. This was 1973 and we were working on real time control systems, not just operating systems. The AWACS was a new airframe. It was a new radar. It was a new computer. The software had never been built before. It was all online, all real-time, and we were under budget and ahead of schedule on every software delivery on that project. There was very little written about it because it was a military project, but AWACS was one of the most successful software projects ever.

The key to it was that in 1973-74, we decided to model everything we did. We modeled all the requirements, we had all the requirements collected, and we modeled them using petri nets. And we had a large conference room that had the onboard software on it, with drawings all over the walls, and when you had a piece of code that

you were getting ready to write, you first took the model- your piece of the model- went with it to an engineering review team and explained how your piece of software was going to work so that it didn't corrupt the rest of the software. Once you explained that, and once you got the approval of the engineering team, you went back to write your code, got your unit tested, and then sent it through the QA cycle.

We modeled everything. And the first time we flew that plane, the only problem we had was that the radar was too sensitive. The software worked perfectly. It was an absolutely amazing piece of work. There were several hundred people writing code and working on those systems, and that's how we controlled the volume and complexity. It was great engineering.

This was in my early computing years, before I went back to graduate school. Consequently, I started to develop the misapprehension that all systems were built this way, i.e. everything modeled up front, requirements always in hand, rigorous engineering at every turn. Yeah, right.

I stayed at Boeing and moved back to their Colorado offices and did work on the analysis team at NORAD Cheyenne Mountain, which at that time was one of the bigger systems in existence. We went right in after the government accounting office had gone in and published their famous NORAD report. Keep in mind that this was the 1970's, the late 1970's. The Cold War was still going on, and NORAD was supposed to be our shield against the godless Soviets, and the GAO report comes out and states that NORAD Cheyenne Mountain and the Air Force has upgraded all their computers, and that these are wonderful computers if you're running an insurance company, but they're not going to do the job for defending the US. Consequently, we went in and looked at everything and did all of the background analysis work and set up what they had to do for the next series of computer performance analyses. That was an interesting project.

I subsequently went back to graduate school and got an MBA at the University of Denver. This was an Executive level MBA, mostly in Operations Research. I focused a bit on ops research, then left Denver and went to New Mexico where I worked in a software startup in Santa Fe for about two years.

Shortly after going public with the startup, I got a call from an old colleague of mine who was at Los Alamos National Labs. He explained that they were going to be doing a lot of distributed computing, that they were going to have to build these models, and wanted to know if I would come up and take over one of the development crews. I said yes, and the result was that I ended up having 80 people working for me who had no process and no understanding of what was going on. One thing we did have, though, was metrics. That's because there was one guy there who had kept a count of how many bugs they had and how much time things had taken. This had all been classified so we actually had about 5 years worth of metrics at our disposal.

The first thing a couple of us did was to look at these metrics. Then we completely reorganized what we were doing, and started putting smaller teams out to handle distributed computing. We had two CDC 204's, which were fairly big supercomputers at the time. And if you think that writing COBOL today was interesting, imagine writing COBOL for a CDC 204, when the only COBOL compiler you could get was built in France. There were simply no U.S. COBOL compilers at the time that ran on scientific computers.

So we had these kinds of issues. We were moving onto VAXs, and we eventually replaced the two CDC 240s with close to 100 VAXs. And we were also getting tools and looking at the relation of databases at that time.

What was nice about being at Los Alamos, as a staff member there, was that we were also Associate Professors at the University of California. So I got to work with a lot of people at UCal. I worked with Mike Stonebreaker and Gene Wong and all those guys who basically invented INGRES. Ted Codd, essentially the father of relational databases, came out of IBM's Santa Teresa Labs. I worked with Ted, and we wondered how and why you would go from relational algebra to relational calculus. This was before Oracle had a really decent engine. And we brought relational database tools into the Lab and wrote some of the first prototyping tools.

We did a lot of prototyping with metrics. We actually put in place the metrics and did the measures. We found that bringing in relational databases, doing reviews, and breaking out separate maintenance groups and test groups improved our productivity almost tenfold. This was in the 1980s. We subsequently wrote papers on it and some

of this information and research was eventually brought forth in our book at the University of Texas.

After Los Alamos I moved back to Texas and worked at several companies, including Motorola. Both my wife and I worked at Motorola. We helped get the software engineering curriculum at Motorola University started and both of us worked together to set up their offshore sites. I worked in India to set up the CMM level 5 group in Bangalore, a development shop that I later had to use myself to build products. My wife worked on putting together shops in St. Petersburg, Tiensin in China, and the Philippines.

At the end of all of this, sometime in the mid-1990s, I put a paper together on offshore costs and what I discovered was that at Motorola, Advanced Micro Devices, and Cirrus Logic (I had worked at all three of these organizations) the ultimate cost for somebody offshore was almost the same as an engineer in Austin, Texas. I could not reduce my cost internally. And there were just too many costs involved in going to India, e.g. getting hardware in, setting infrastructure up, etc. The costs in Russia were really wonderful but there were other problems over there.

In Russia, I had a room full of PhD's. Really smart guys but you had to manage the hell out of them, because if you think American programmers are cowboys, Russian programmers are even worse. What we didn't realize was that, in the fine print, they got to keep the hardware at the end of each project (we were working on a project by project basis). I had purchased eight thousand dollars worth of Sun workstations for these guys, the project was over and we were getting ready to start the second project, and I get a call from the project manager asking, "So, when do we get our new hardware." "You've already got your hardware," I replied. "Oh no, we've already sold that hardware. When do we get the new hardware?" What they were doing is selling the hardware on the black market to make up for the fact they were getting paid such a small salary. That was a tough lesson to learn.

In any case, it was around this time, late 1990s, that we started Athens Group.

**CAI: Could you tell us a little bit about what you do at the IEEE?**

**DON SHAFER:** At the IEEE I am the Editor in Chief of the Computer Society Press. The IEEE is the International Electrical and Electronic Engineering Association. We've got a couple of hundred thousand members worldwide. The computer society is one of the largest societies there. I'm also a senior member of the IEEE.

I've been the Editor-in-Chief for two years. What we're doing right now is building a set of how to books on using IEEE standards. We're trying to build some books around this, some handbooks that make them useful for people.

**CAI:** You wrote a very highly regarded book on Project Management, along with your wife Linda and also Robert Futrell- *Quality Software Project Management*. There is an anecdote from one of the book's opening chapters that I found intriguing. You begin by stating that, "In October 1968, 50 software engineers from 11 countries attended a NATO science committee meeting in Garmisch, Germany." Could you explain for us what came out of this meeting, why it was so important, and what its impact has been on the software industry?

**DON SHAFER:** The number one problem addressed at that meeting was that there were simply not enough programmers in the world. The second problem addressed was that the programmers we did have had no processes. The meeting itself was very focused on the military and the needs of the military, but they were also concerned about banking and some of the other major industries.

What's amazing about this is that the year was 1968. It hadn't been that long before that Thomas Watson, Sr. had remarked that we only needed 10 computers in the whole world. The software field had simply come into being too quickly and, as a result, there were no formal software engineering practices to refer to. In fact, many places didn't even have computer science curricula. And so the outcome of 1968 was that we got involved with the problem of how to create more programmers and how to get process for these programmers.

But the really ironic thing is this: I've still got a copy of the original minutes from that meeting and it reads like the stuff we talk about today. "Nobody's got good

requirements." "Nobody does metrics." "Nobody knows how to measure productivity, everything's always late, and over budget." Whenever I read through this I think, "My God, that was almost 40 years ago and we still haven't learned a damn thing!" And we haven't. We really haven't.

**CAI: What do you attribute this to?**

**DON SHAFER:** I hate to say it but much of that fault must be shared by our customers, because our customers simply do not understand what it takes to get software out and to get it right. Moreover, most software consumers in general have been so desensitized by Bill Gates and Microsoft, by the fact that stuff routinely doesn't work, that they are quite willing to work around problems when they do inevitably arise. But you can't do that. You can't do that in security, you can't do that in banking. I don't want Wachovia Bank to take some cavalier attitude about my electronic fund transfers. I don't want Schwab to. I certainly don't want that attitude to be prevalent when I go into the hospital.

The remedy is simple. Customers have got to learn how to stand up and say "no!"

**CAI: Much of your work revolves around project management. Why in your opinion is good project management so important in software development and maintenance? Why do you feel it's worth paying for?**

**DON SHAFER:** Software development has got to be treated like a project. Why? Because it's got a beginning and it's got an end; because it has a budget and it has resources that you place against that budget.

Think about it this way: would you ever consider building a road or a bridge without project management? Let's say you wanted to build a new turnpike. And you're going to have lots of overpasses. Do you tell yourself that, when it's time to put an overpass in, that you're just going to get a bunch of guys together and throw some forms up and pour some concrete and make an overpass? If that sounds absurd, why do we do it with software? Why do we do it with payroll systems and healthcare systems? One of

the worst cases on record right now, one that is going to go down in infamy, is the FBI's team case system. After 9/11 they finally realized the importance of getting all of their 286s and 386s integrated into one case system. Consequently, they went out and they spent 200 million dollars with SAIC who managed to fritter it all away. They didn't have a good project plan, they didn't conduct reviews, and now they've got yet another piece of software that doesn't work. And they have no choice but to throw it all away and start over again because they didn't have proper project management in place the first time.

At Athens Group we manage everything we do as a project. Even internal work is considered a project. We have estimates and reviews, we measure things, everybody turns in a time sheet, and we know exactly how many hours people spend on their work. If you're not doing things like this, how do you expect to be able to manage a project? How do you expect to manage a project when you don't know what your outcome is going to be? Not only that, you will very quickly come up against a classic software problem- you will find yourself at the "end" of a project, 90% done, but you will have already spent 100% of the budget. And it's going to take the second 100% of the budget to get that last 10% completed because, the truth is, you're not really 90% done. Without good project management, you don't really know where you are.

**CAI: What are some of the biggest misperceptions that revolve around project management?**

**DON SHAFER:** The biggest misconception is that there is no value-add. There are so many software organizations that cling stubbornly to this belief that professional project managers bring no value to an organization. And that is unhappily untrue; because any time you look at projects that are successful, you will always find a project manager component there. Projects that have no project manager are generally not successful. There may be a technical lead, a team lead, but you will not find a project manager. Technical leads and team leads are focused on getting out the technical product. A project manager is looking at the entire picture. The project manager also has an important role to play as one of the primary interfaces with the customer. The last thing in the world that you want is for your technical lead to be pulled away from

getting good technical product out the door because he is too busy talking to customers.

So the value is there. But the only way to show the value is to have the metrics. And you won't have the metrics at your disposal unless you are a relatively mature organization. That's kind of the paradox.

What usually happens with most projects is that, when it's finally over, after the last death march, everybody on the project goes out and has a beer and pizza lobotomy and they simply forget about it. And then they go out and do the same thing the next time. The definition of insanity is doing the same thing over and over again, expecting a different result each time. Without project management, your organization is truly insane.

**CAI: In your book *Quality Software Project Management*, you outlined 34 key competencies that every good project manager should know. Could you discuss some of these briefly? What are some of the most important?**

**DON SHAFER:** To go through all 34 would take about 52 weeks. But let me share with you some background.

Those of us who put that list together- people like Herb Krasner, Bob Futrell, my wife Linda Shafer, Joyce Statz, and Bill Curtis- we started building this program in 1991 at the University of Texas and it was aimed primarily at developing good software project managers. Since then, we've had hundreds of people go through these sessions, and it's still going on now. There's an online component to it, too. And the book that we wrote- *Quality Software Project Management*- is really the textbook for the class.

But regarding the 34 competencies, the key here is that the competencies fall into three groups: People, Process, and Product. And those three key areas are what you focus on. Don't underestimate the importance of people skills. You can't do project management by sitting in your office, keeping the door closed, and only responding to e-mail. You've got to have people skills. You've got to know how to be a change agent. You've got to know how to work with teams. You've got to be able to build teams. I

think one of the key areas is not technical at all. It's understanding how teams work, understanding team dynamics. You almost never build anything by yourself anymore.

I should probably apologize to everybody, because I've been doing this for 34 years, and those of us like myself and Bob Glass who've been there since the beginning, since the late 1960's and early 1970's, we basically did all the easy stuff. All the easy stuff is done. You guys can't do it any more. We fixed it. The easy stuff is gone. We took care of it so now you've got all the hard problems to deal with. And the hard stuff requires teamwork. And you've got to understand team dynamics, the ideas of forming, storming, norming, and performing. For teams, this is very important.

When the team first forms up, everybody is happy. Everything is great. But within a short period of time, everybody hates each other and wants to cut each other's throat. They can't stand it. They hate each other, and it takes a leader to go in and get them to start working together again. Then you've got to norming, and the norming is the setting of expectations. Then you finally get to the performing. And that can take a short period of time or a long period of time. I've actually seen teams that have never gotten out of the storming. They just canceled the project and started over again with a new team. And I've also seen some teams that get all the way through within a couple of weeks. And they're performing at a very high level, because they've figured out how to work together before.

So I think a very critical area within all of the 34 key competencies is the ability to get people working together on teams. The rest of the competencies are things we already know: collecting metrics, conducting reviews, doing prototyping, having a project manager, doing real product management, having product definitions, etc. The real secret, however, is getting the teams working right, because this is where your product is going to come from.

**CAI: What does an organization need to have in place from a process perspective, before they can expect to derive any significant benefits from their project management program?**

**DON SHAFER:** There are probably three that are critical, three processes that an

organization has to have. First, an organization has to have a lifecycle. If there's no lifecycle, you will have no roadmap for building your product. You have got to have a lifecycle in place. You've also got to have reviews in each of the phases of your lifecycle. And if you've got marketing people involved, you should probably get them involved in these reviews, too, at least in the beginning, in your concept phase. The third thing is you've got to put in place is some kind of a metrics process. You can't manage what you don't measure. And if you're not going to measure it, it's probably not even worth doing.

**CAI: Are there any specific metrics that you feel are particularly important for managing projects properly?**

**DON SHAFER:** You've got to know the amount of effort that goes into each phase. That means that you've got to keep track of effort metrics, whether in terms of hours, days, or whatever. That's the major one. This will tell you how much time you've spent in each of your phases.

You also need to get some kind of metric on rework. And to do that, you've first got to decide on your unit of measure. Is it function points, or is it feature points, or is it a UML-oriented case point, or is it lines of code? My own preference is lines of code.

I could use case points, or feature points, or I could use function points, but at some point in time when you start collecting errors (e.g., how many errors escaped, how many errors were on your final project, etc...) you've got to know your lines of code. I can give you some specific examples. We built the infrastructure for the Dell factory outlet. A month before we started this Dell was the biggest personal computer company in the world. When they were still a screwdriver shop, just putting components together, they started getting all this stuff back so they decided it would be a good idea to put up a bricks and mortar factory outlet. Eventually, they wanted to get this outlet online, but they couldn't put it online in their Dell premier area, which handles millions of transactions a day, so they had to have their own web area. They designed the look and feel of their website and we built all the background software using Websphere, Visual Age, and lots of Oracle interfaces. It ended up being about

three quarter of a million lines of code with everything that was in there.

We had processes in place. We knew exactly what it took and how long it took to do the requirements. We knew exactly what it took to do all the pieces. Then we instrumented it for any problems that occurred. Consequently, we ended up with just four major problems that were due to misunderstandings in the user interface. After four years we've only had six lines of code that we've ever had to change. That's it.

These are metrics you have got to have.

**CAI: Once organizations start gathering these kinds of metrics, what do they need to know about the processes that go into analyzing it? Ultimately, the metrics are not really going to be very useful unless you know how to analyze the data and are able take some kind of action with it. What are your thoughts on this?**

**DON SHAFER:** One of the things you've got to understand is the type of work you do. Is your organization a traditional IT shop where you've got maintenance work, new development work, and enhancement work that's going on all the time? If so, you will want to categorize your metrics into these areas. How much time does it take, for instance, for you to do a complete a maintenance request? How many lines of code do you change when you do a maintenance request? Once you identify these metrics, you can start putting stakes in the ground around them.

We had a problem several years ago related to code that was in maintenance. There was a lot of dead code. That means that when you make a change in one place the change is more likely to break something somewhere else. To deal with this, we used McCabe's cyclomatic complexity metrics. We ran all of our code through this and got back a number back, a complexity number.

This complexity is a reflection of a lot of things: your branching structure, your control structure, the languages you are using, etc. We produced this metric for all of our code. We had thousands and thousands of subroutines and different types of modules in our code base but we gave each one a number. We then put this number into our

front end, into a configuration management system.

To demonstrate the practical value of this, what if I am a programmer and I get a call at three o'clock in the morning that says, "My god, our EDI system is down, and we can't do an electronic data integration, we can't get our information in to get our accounts receivable going. You've got to fix this!" At three o'clock in the morning there's no change control board. And you've probably got the authority to make the change anyway. So you check out the code, make the change, and check it back in. But there's a script that runs first, before you can check it back in. And this script evaluates your code complexity. And if you've made this code more complex than it was before you checked it out, it will not allow you to check it back in. This forces you to reevaluate things, remove as much of the dead code as possible, take out the stuff you branched around, take out that crap you cut and pasted in, and fix it good and proper so that your changes don't wind up breaking something else down the line.

With this small exercise alone you've just solved a big maintenance problem. It's actually an enormous maintenance issue. And you solved it very quickly and very cheaply. It takes no time to run your code through this complexity calculator. You can probably complete it all in a couple of days. It is a no-brainer.

**CAI: And what is your advice for people who find themselves coming up against human resistance to a metrics program? How do you deal with this as an effective project manager?**

**DON SHAFER:** One thing you can do is make sure that the organization has a vision and a set of goals that they are all walking towards. Also, and this is very important, when you do the measurements, you've got to make sure that they are anonymous. You've got to take people's names out of things. We always render the data anonymous. If you dig far enough and you look at it right, you will know who's working on what. But when you roll it up to an organizational level, you won't. You really don't want to have people start using the data to point fingers at each other.

One of the worst things that I ever had happen was within an aerospace company. We were trying to get them to put processes in place. And we had finally managed to put

some metrics in the ground. And we were in this one meeting where we were just getting ready to roll the metrics out, and all the stakeholders were in the room, along with some of the managers, and everybody was very excited. In the middle of our presentation, one of the managers raised his hand and said, "This is really great. I just want to say this is some of the best stuff I've ever seen. Now I'm going to have data at my disposal when I give out raises next year." And he was joking, but it was a bad joke, and it killed 8 months of work. We never rolled it out. All of a sudden everybody in the room looked at me and said, "You lying sack; you're going to use it for performance reviews? We're not participating." And they didn't. The client had spent hundreds of thousands of dollars on consulting, on putting process in place, on collecting initial data, and one careless remark killed the whole program.

**CAI: What additional reading material might you be able to recommend for those among us who are aspiring to be project managers? What kinds of skills should they be actively seeking to get under their belt in order to succeed and remain competitive in this area?**

**DON SHAFER:** The Project Management Institute website is incredibly valuable. The Software Engineering Institute at Carnegie Mellon has got an enormous wealth of information, too. One of the better ones is the Project Managers' Network, which is a website that was developed by the Department of Defense. It has an enormous amount of data.

Certainly your own newsletter, the IT Metrics and Productivity Journal, is very important; in particular, the interviews and the web based research repository. I would definitely recommend that people make an effort to read these interviews. The content is outstanding.

For those of you who want to become a professional project manager, I see more and more people pursuing their PMP certification. That stands for Project Manager Professional. It's run out of PMI and it's a very worthwhile union card to pick up. While earning your PMP certification, you will learn a lot about overall project management.

Some of the other approaches are just finding people who've done it, reading some of

their books, and going to seminars. There are seminars all over the place these days. You could always contact a local Software Process Improvement Network (SPIN). They focus on a lot on project management.

Look locally, but look on the web first. You can go out and buy tons of books and you can go buy our book if you want to, but I would really start with the web first and find out what you're interested in doing, because there is just so much information about this on the web.

There are also a lot of tools that you can download. There's Sourceforge.net as a really valuable web site for open source software. There's Risk Radar as a free risk management tool from SPMN.com and there's all kinds of project management tools. And while we are on the subject of tools, if you're a Microsoft-centric shop, it's always good to learn Microsoft Project. Microsoft Project is useful because it's a better drawing tool than Visio. It's also quick and easy to use, but it's still not an industrial strength tool. I wouldn't seriously invest in tools until you really understand what you're doing. You will turn into a mouse driver. The real key is walking around and interacting with people, finding out what's going on.

Questions? Suggestions? Comments? Please contact the IT Metrics and Productivity Journal Editor at [michael\\_milutis@compaid.com](mailto:michael_milutis@compaid.com)