



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

**Focus on Dr. Elizabeth K. Clark**  
**President, Software Metrics, Inc.**  
**A CAI State of the Practice Interview**  
**December, 2005**

**Biography of Dr. Elizabeth Clark:**

Dr. Elizabeth Clark has been involved in the practical application of measurement for predicting, controlling and improving software process and product quality since 1979. She is the President of Software Metrics, Inc., a consulting company she co-founded in 1983. Dr. Clark is a primary contributor to *Practical Software Measurement*. She is a certified PSM instructor and has conducted numerous PSM training classes and workshops within the United States and Australia. Dr. Clark was also a principle contributor to the Software Engineering Institute's (SEI) core measures.

Through her affiliation with the Institute for Defense Analyses, Dr. Clark has extensive experience in performing independent cost analyses for government clients. Her experience covers a range of weapons platforms as well as large information systems.

Dr. Clark received her B.A. from Stanford University and her PhD from the University of California, Berkeley. Our interview between Elizabeth Clark and Michael Milutis, the IT Metrics and Productivity Institute's Executive Director, took place in September 2005.

◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆

**CAI: Could you tell us a little bit about yourself, the path your career has taken and what you are working on today?**

**CLARK:** My training was in a branch of experimental psychology, cognitive psychology, which is the study of how people think and learn and remember things. And instead of doing experiments with rats I did experiments on college freshmen. I did my undergraduate work at Stanford and got my PhD from Berkeley, so I was really being trained for an academic career. At the same time, however, I had taken a couple of programming courses, and was really getting interested in computers. There were not a whole lot of academic jobs at the time, so I started thinking that a career in computers might be interesting. It was also obvious to me that computers were the wave of the future. This was back in the late '70's.

There was a group back then at General Electric in Arlington, Virginia that had been started by Tom Love. Tom eventually left and then Bill Curtis, who later went on to run the process group at the SEI, came in to head things up. It was Bill Curtis who hired me.

He was looking for a cognitive psychologist to do experiments with programmers. In those days, the cost of hardware had always overshadowed any other kind of computer related expense, but writing and maintaining code was starting to get much more costly. Consequently, people were getting more interested in finding ways to make programmers more productive with fewer errors. To this end, I was charged with the task of putting together some data, of conducting some empirical research experiments, on what might make programmers more effective. This was how I got started. Bill Curtis actually left 9 months after I was hired and consequently, I took over his job and headed up the group.

After 3 years I left and started my own company, Software Metrics, Inc. I've been doing this ever since. I essentially went from studying individuals to working on project measurement and cost estimating. I actually spent a lot of time with the Institute for Defense Analyses in Alexandria, Virginia, from 1984 to 1996, doing cost estimates on large weapons systems. For example, the B1-B Bomber was a project I worked on. That was a big one. I had to work through all of the different parts that go into estimating the cost of software for that project- the whole set of lifecycle costs.

I also do a lot of work with the FAA and other government agencies. For the past few years we've been implementing an enterprise measurement program for Customs and Border Protection. I do some commercial work, too.

**CAI: Much of your current work is focused on estimation. Why is the estimation of cost and schedule so important to software project success? Why is getting this right is so important to the business?**

**CLARK:** I think that people by their nature tend to be optimistic. We tend to think of what it's going to take to get something done, and we never anticipate the things that are going to go wrong. So if you already have unrealistic plans to begin with, you can really get hurt. A lot of projects, if they haven't estimated well, end up dead in the water from the very beginning.

I'm a big believer in, and also a user of, parametric cost models. Parametric cost models can be used to compare against what other projects have done with similar complexities and similar applications. This can help you get a more realistic understanding of the cost and schedule risks facing your project.

**CAI: What are some of the biggest challenges that organizations encounter with software estimation? How does this differ for large organizations like military organizations, as opposed to small to mid-size organizations?**

**CLARK:** The biggest challenge is always the estimation of size. I spend 90% of my time just trying to get a handle on size.

Most cost models run off of lines of code, but I've never seen a project that was straight new lines of code. There's always some reuse and some modification. Consequently, models make use of something called "equivalent" lines of code. So if you're reusing some code or modifying some code, you always have to be asking yourself what the equivalent

would be in terms of new lines of code. You always have to find that common format. It has gotten much more challenging to do this in a consistent way now that we have programming languages in which people really aren't writing straight lines of code anymore. And I would say that this remains challenging regardless of the size of the organization.

**CAI: For organizations that want to get this right but maybe are not as sophisticated as some of the military contractor sized organizations, what kinds of questions should they be asking themselves? Are there any first steps you might recommend to help them get started?**

**CLARK:** One of the things they can do right away is to start collecting data on themselves in order to better understand the nature of the work they do. That means getting some consistent measure of what's being done in terms of size and scope, effort, and calendar time (i.e., duration). It's also important to make sure you enforce consistent measurement across projects, because if you try to get measures from different projects, everybody is going to have their own definitions. By way of example, if you're measuring effort in terms of staff months or labor hours, are you including administrative staff or project managers in this category? Are you including overtime? These are the types of inconsistencies that you will encounter across projects.

The second thing that organizations should do is to start using cost models. Some of them, like COCOMO, are available for free (you can get COCOMO from USC). However, whether you get your cost models for free or spend a lot of money on them, the important thing is simply to start using them and to start calibrating them with you own data. Just taking any cost model and plugging it into an organization will not be nearly as effective as calibrating it to local data and local conditions.

**CAI: Could you define cost estimation models for us and explain the basics of how they work?**

**CLARK:** A cost estimation model simply relates the amount of work to be done, which is generally expressed as either function points or source lines of code, to the amount of calendar time and human effort it is going to take to complete. The model works because time and effort are related to size. That's the basic equation. And once you have effort, you can get dollars.

There are also certain parameters that are used to adjust the estimate, which have to do with the complexity of the software, the capability of the people, the experience of the people, the application domain, etc. This is what makes the model "parametric." Other parameters include tool support and the quality of the tools. Anything that can impact productivity, effort, and duration is a potential parameter. A model is just a way to input what you know about a project in order to get an estimate in terms of time and effort.

**CAI: Could you give us a quick inventory of some of the common models and**

**approaches to modeling, and maybe shed some light on what distinguishes them from each other, and how organizations might determine which model or technique they should be using?**

**CLARK:** Probably the simplest method and the one we use most often is expert judgment. It's really just bringing people together to guess based upon similar applications or work that's been done before. That is really the least formal approach. At the other extreme are parametric cost models, e.g. COCOMO, COSTAR, SLIM, SEER-SEM, etc. These models are primarily differentiated by their underlying assumptions.

You really do get what you pay for with models. As they get more expensive, they tend to have very nice interfaces. They also tend to get much more sophisticated in allowing you to do "what-if" analyses. For example, what if the size turns out to be 20% more than what I estimated? What if we need to compress the schedule by 30%? What if we end up with half the staff that we're anticipating? With this type of functionality, you can very quickly run through a lot of different risk scenarios.

One of the things that people need to realize with estimates is that they can be associated with probability. A lot of estimates, for instance, might give you 50% probability, which means that half the projects will get done in less time or less money and the other half in more time or more money. Consequently, if your tolerance for risk is not very high, you might want to shoot for an 80% or 90% probability level. Sophisticated parametric models will very easily let you do that.

**CAI: Is there a significant amount of customization that takes place when you're working with these types of models or is it more or less off-the-shelf?**

**CLARK:** You can just plug them into your computer and use them as-is. However, I'll say that with two caveats. These models become much more accurate when you gather up your own local, organizational data on size, time, and effort. The other thing I will say is that size itself is a whole tricky business, because I've never seen a project that consisted purely of new, third generation lines of source code, which is what these models assume. So if you are building active server pages for a web-based system, and you're using languages like Visual Basic, it's going to be a whole different paradigm. Trying to map that back to lines of code isn't going to be very straightforward.

**CAI: How does one deal with this?**

**CLARK:** You have to focus, once again, on consistency of definition. You really have to characterize the work you're doing as either lines of code or as equivalent lines of code. That's the short answer.

**CAI: Estimation is fairly interrelated with process and measurement. In light of this, what must organizations minimally have in place on the process side and the metrics side before they can expect to see any returns on their estimation efforts?**

**CLARK:** That's just a great question. I think it gets at the heart of what people struggle with in this industry. What I'll tell you is that data reflects the process that produces it. Data comes from a process. So if processes are ad hoc, if they tend to be chaotic, then your data will be messy. And if every project does something different, and uses different processes, then it is going to be very hard to compare what's going on. With standard processes in place, however, you can really start collecting data that will enable you to make meaningful comparisons, and your estimates, in turn, will get better over time.

**CAI: It really seems almost impossible to have a discussion about any one of these three topic areas- standard processes, measurements, or estimating- without coming back to a discussion of the other two.**

**CLARK:** They are completely inter-connected. This makes it particularly challenging to give advice to people on how to get started when they're coming from a zero state, i.e. from a state of not having anything standardized, not having any kind of process or any kind of metrics any kind of data collection.

**CAI: What do you tell an organization that is in this kind of a situation? How would you get them started with all three of these things at once without it being too much for them to bite off?**

**CLARK:** Even when you just walk into an organization that is at level zero, you can still apply models, you can still collect basic data just by interviewing project managers on recent projects. You won't get perfect data, because nothing has been collected or retained, but you can still start getting some idea of size and scope and of effort and schedule. It might mean getting function points from documents; it might mean counting source code from existing programs. You can also start calibrating models, and if you don't want to go the model route, you can start building basic estimating relationships whereby size, effort, and schedule get related. This is all that models are doing, it's just that they do it within a mathematical equation. If you don't go the model route, you can always just start building up some relationships. That's at least a place to start right away, with what you have.

Regarding getting started with measurement, all projects have existing data around. All projects collect status and track it in some way. They may all be doing it differently, they may be using simple Excel spreadsheets or Microsoft Project, but there's always some existing data that you can grab and that you can leverage to create useful graphs or indicators to start providing visibility into what your organization is doing. And you should start with size. As long as you have size, you can start measuring productivity and quality. And once you start attaching numbers to these phenomena, you will have baselines to improve upon. You will also start getting some management attention into these issues. That's important because in order to keep improving, you are going to need sustained attention at all levels.

All of these measurements will be very noisy until there's some real process improvement. However, for getting started, you can really just pick any one of these

points- process, measurement, or estimation- to focus on. As the processes improve, the estimates will get better because you will have definition and standardization behind things. You will also get much better visibility into what's going on. This will help you figure out what processes to improve, for the future. It really all hangs together.

**CAI: What advice would you have for organizations that, for whatever reason, are still struggling with this?**

**CLARK:** Keep working at it. I would also strongly suggest that you try to make measurement as easy as possible. You can do that by gathering up data that already exists, as opposed to implementing it as an overhead function or creating a lot of new things that people have to do just for the sake of measurement. The more data that you can pull from what already exists, the better off you will be.

Tools are important, too. They are important because they can give you access to the data in real-time. And that will help you manage and communicate better.

**CAI: Should an organization's approach to development-based estimation be the same approach used in maintenance? How would you differentiate these two areas?**

**CLARK:** I think maintenance is another can of worms. People really struggle with it. The reason maintenance is really hard is because the amount of effort that's involved in maintaining software is not as strongly related to number of lines of codes, per se. And there's a lot more analysis that often has to be done just to determine the impact of the change. For instance, if you have systems that are deployed at many different sites, there's just so much involved with new releases that in some cases there can be a huge amount of maintenance work that is related in only a very minor way to the actual number of lines of code that might be changed. And that is very different than development.

In my experience, models don't do as well with maintenance. They do try to address the issue, though. COCOMO, for instance, looks at the amount of redesign that is done, the amount of recoding, the amount of retesting, etc. However, it's been my experience that the cost to maintain is just not very strongly related to the amount of code that's being changed.

**CAI: One thing that is striking is that there is all of this research and publishing that's done in the area of new development, and yet we hardly ever see anything written about maintenance. Has your work ever lead you directly into this area of research?**

**CLARK:** I actually did a study with several other people, about 8 years ago, that looked at maintenance organizations within the Department of Defense. It was very interesting because these folks- the maintenance groups- got left with the decisions that were made

during development. These were decisions that were originally made to get something out the door quickly, but that made the system very hard to change or maintain. And the people maintaining it were really stuck with those decisions. They had very little voice in the initial decisions that were made relative to architecture or design of the system. Yet they were the ones who had to live with the consequences.

Where we see this a lot now is in the whole world of commercial off-the-shelf software (COTS). Within the government, at least, there's been a real push towards going the COTS route, under the assumption that you can get systems out the door cheaper, faster, and with less risk. But what happens is that the decision makers in these cases aren't thinking about maintenance. And that has a lot of implications, especially when it comes to risk. I've seen systems, for example, that have had 150 or more components from different vendors, all of which are evolving in a way that is determined by each vendor. This will really increase your risk levels, since you are not only dealing with interoperability problems whenever new functionality gets added to individual components, you also have to deal with vendor support for various components being dropped after a certain period of time. At that point, your choice will be to either upgrade, if you want to continue getting support, or you take your chances. And if you have a safety critical system, and I know this very well from working with the FAA, there will be huge amounts of analysis required on critical COTS components by the maintenance people in order to determine the impact of going with an upgrade versus not going with an upgrade.

**CAI: So you feel pretty strongly that the COTS approach to developing software increases maintenance overhead in the long run**

**CLARK:** Yes, definitely. I've actually done a lot of research in this area, working with Barry Boehm and some of the folks over at USC on a model called COCOTS. COCOMO is the constructive cost model and COCOTS is just a member of that same constructive family. COCOTS is for COTS based systems.

The FAA was a sponsor for this research. And the reason that the FAA was a sponsor was because of the life cycle implications. What I did in the course of this work was to research almost 30 different projects, from development through maintenance, and the thing that I consistently heard people say was that these COTS-based systems were definitely more expensive to maintain, in the long run. I think that was a real surprise.

**CAI: Are there any practical solutions that you might be able to recommend for organizations that, through their use of commercial off-the-shelf software, may eventually face some of these very same challenges?**

**CLARK:** COTS problems can be preemptively addressed in the architecture phase. One of the projects I worked on where this was done right was at NASA's Hubble Space Telescope program. The NASA folks right up front identified the critical components they had in their COTS-based ground support software. They then architected their system to be able to swap out any commercial component and put in another, in such a way that the rest of the software would be protected. That was the best approach that I've seen so far. They were really thinking about the future at NASA and they built a system in which

they knew they would be able to either upgrade or switch vendors. You have to keep in mind that sometimes vendors just go belly up. If you don't plan carefully, your project can get caught up in that as well.

**CAI: For people interested in all of these subjects, could you maybe recommend some additional reading?**

**CLARK:** I co-authored a book several years ago that I would recommend to anyone who finds this interview interesting and would like to read more along the same lines. The title is *Practical Software Measurement*.

If I had to name a few classics I would start with Gerald Weinberg's *The Psychology of Computer Programming*, which came out in the late '70's. Weinberg argued that programming is a human activity. To this day you still have people talking about software engineering as if it were a pure engineering activity, but it's actually a very human, cognitive mental activity. I find this particular dimension of programming very interesting.

There's also a book Barry Boehm wrote called *Software Engineering Economics*, that I believe came out around '82. It's called the "blue bible" (it's got a blue cover). I still go back and read that book from time to time. That is really the best book on software estimation I have ever read. I think it still holds up well to this day.

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