



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

**Focus on Lawrence Putnam**  
**A CAI State of the Practice Interview**  
**September, 2006**

**Biography of Lawrence Putnam**

Lawrence (Larry) Putnam is a renowned authority on software estimation and measurement. Larry was the first recipient of "Freiman" award for sustained superior performance in parametric estimation covering a ten-year period. He is the founder and has been President of Quantitative Software Management Inc. since 1978. Before establishing QSM, Larry had over 26 years of experience in software and hardware resource planning, estimating and allocation. Over the past 20 years, Larry has conducted extensive research into software cost estimating techniques using operations research modelling techniques to determine significant cost, schedule and reliability drivers. He has extensive experience in data collection and analysis techniques. Besides publishing over 30 papers on the subject of software estimation and measurement, Mr. Putnam has authored/co-authored many books. Our interview between Larry Putnam and Michael Milutis, Executive Director for the IT Metrics and Productivity Institute, was conducted in May of 2006.



**CAI: Could you tell us a little bit about yourself, your background, and what you are working on today?**

**LARRY PUTNAM:** I'm a 1952 graduate of West Point and a retired Colonel with 26 years on active duty in the army. During my service time, I studied at the US Navy post graduate school and in 1961 received a Masters degree in statistics. My course of study was nuclear effects engineering. This led to an assignment at Sandia Base, New

Mexico where I had to do a number of blast calculations. I trained as an engineer in the slide rule days, and this was just at the end of the slide rule days.

Next door to the base the Sandia Laboratories had a brand new best in class computer and they were willing to share it with the people stationed on the base. I said to myself, "Hey, here's a great chance for me to get all these blast calculations done, and even learn how to use the computer." So I signed up and took a course in FORTRAN programming during lunch hour. When I finished, I had received a nifty little certificate of completion. Like a good army officer I popped it in the mail and sent it down to my personnel records in the Pentagon. I then promptly forgot about it until a couple of years later when it was time for me to do some duty down in the Pentagon. They said "Larry, we don't have any jobs down here for you now in the nuclear area, but we notice that you're qualified in data processing." They had seen that FORTRAN programming certificate. I was subsequently asked to come down and help run the army's data processing budget.

Six or eight weeks later I went to the budget table with the analysts from the office of the Secretary of Defense and they looked at all of our computer leases and software - about 100 million dollars worth. They went through each of these programs and came to something called SIDPerS which was for a standard installation division personnel system. We had completed the development in about four years and it had gone active at about 90 different installations around the world. They then looked at the budget for the next five years and said "We noticed you finished this and had 116 people working on it. Next year you're asking for 112 people and then for the five following years you want 96 people. What are they going to do?" I didn't know the answer, so I turned to the civilian on my left and the civilian on my right, who came into the Pentagon at the same time the first computer did, and they mumbled "Hey, we don't have a decent answer." So the analysts said, "Well look, let's adjourn. You guys go out and find the answers to this little problem and we'll convene again at 9 o'clock tomorrow. Give us a good answer and we won't have to take 10 million out of your budget." The next morning, we convened again at 9 o'clock and by 9:30 I had lost 10 million dollars of the Army's budget. I had to spread that pain amongst a bunch of people who were, needless to say, not very happy with me.

Interestingly enough, about two or three weeks later I was in the Pentagon bookstore when I spotted a little 69 cent paperback on the get-rid-of-it-quick counter, and it had a chapter in there on managing R&D projects, written by Peter Norton and the IBM Poughkeepsie development lab. He had traced some curves which I subsequently called Rayleigh curves, after a famous British physicist, and he used those for formulating staffing profiles for R&D projects, some of which were in software. The tie-in was that they looked remarkably like some of the budget profiles I had been working with. I bought the book, ran back, applied the math to about 50 large Army IT projects, and as a result we were able to get them all under control in about two months. When I developed this a little further we came up with some algorithms. I thought this was a real big breakthrough.

A year later we had some follow-up budget hearings of the same nature in which we were faced with a budget cut again, a rather big one. This was at the conclusion of the Vietnam War. We used the material that I had developed a year before and as a result were able to persuasively conclude that if the budget cut hit us, we wouldn't be able to develop another software project for three and a half years. As a result of my method, we were able to respond quickly and we also came back with decent answers and this saved our budget.

Shortly thereafter I retired and went to work with General Electric in Arlington. At GE I was able to test these ideas on a number of projects with some success. This gave me the confidence to bring these ideas into a commercial environment. Consequently, a couple of people from GE and I started QSM - Quantitative Software Management - and we proceeded to build the first version of SLIM, which at the time stood for Software Lifecycle Management. It was our own proprietary software estimating system, and we built it to run on dial-up time sharing computers. Later, we migrated SLIM to run on the HP desktop machines. A year or so after that the IBM PC came out, and we quickly adapted it to run on that, too.

Over the next 28 years, we built many more versions of SLIM (which we now refer to as SLIM Estimate). There is SLIM control, which is used to manage ongoing projects, SLIM metrics, which conducts a productivity measurement on a portfolio of software projects, and SLIM MasterPlan, which brings together a whole portfolio of projects so

that you can see the overall spin profile of the portfolio.

These tools help us determine the size, time, effort, staffing profile, expected reliability, and risk associated with projects. They give us the capability to plan and manage the business aspects of software development and they help us overcome the budget, schedule, and quality problems that are endemic in our industry. In short, we have a way to quantitatively manage our software development projects and to bring them in on time and within budget. It's a totally automated process for the entire lifecycle.

That's what we do. We do it well and we have best in class products. We keep them that way and we've been able to transfer our skills and knowledge to our clients so that they can become much more effective software builders, too.

What are we working on now? We are working on easier ways to interface our tools to other people's tools. They often have data there, and they want to move it into our tools and then move the results back out. We probably spent the last four years or so working on this issue. We're also starting to work on web-based estimating ideas so that people can come in and get quick and dirty answers. That's something we expect the public might see in perhaps a year or two.

**CAI: You mentioned the problems we have in our industry with schedule and budget overruns. You're of course familiar with the Standish Group study. The Standish Group reported several years ago that 70% of all software projects were coming in over budget, over schedule, or not at all. What do you attribute this to?**

**LARRY PUTNAM:** Software development has been plagued with this problem right from the outset. In fact, this is why I originally got involved in trying to estimate software projects - to overcome the overrun and slippage problem. Interestingly enough, the numbers you cite from the Standish Group have hardly changed from the time I got started. I used to quote similar percentages in the presentations I gave nearly 30 years ago and apparently, no improvements have been made over this time period! Of course, we ask far more of the products we produce today than we did 30 years ago. Nevertheless, what I can conclude from all of this is that progress has

barely kept pace with our demand for bigger, more sophisticated products.

But to answer your question: why the fundamental problem with overruns and slippages? I would say the answer can be found, primarily, in management's unrealistic expectations; specifically, business pressures that come from management to get something quickly to the marketplace, something good with lots of features. Management, for instance, will routinely dictate that a software product must be ready in X number of months, without consideration for how much functionality has to be created.

**CAI: For organizations that are at Level 0, so to speak, in terms of their estimation abilities (or in terms of their sophistication in dealing with management) do you have any advice to offer? How can such organizations start making a difference right away?**

**LARRY PUTNAM:** In order to be able to do credible estimating, you need to know what your development schedule is for software, because a software schedule is like water. It's incompressible and everybody thinks you can mold it any way you want. But you can't. The schedule will always be very dependent on the size of the project you're undertaking and on the productivity capability of your organization. And if neither the management nor the project staff has a measurement program in place for determining these things, then you are really going to be behind the eight-ball. You are not going to even come close.

A table-banging manager will say, "I have to have this project in 12 months, now get cracking." He is not going to be successful and his organization is not going to produce. It's simply a case of unrealistic expectations without any underlying body of explicit software development knowledge to guide the process.

As far as advice is concerned, my answer is to start a real process productivity program. Measure it, and let that be the basis for a good estimating control system, one that you apply continuously throughout your development efforts. You'll know where you are. You'll know and be able to measure the payoff from your improvement program. Management will be able to manage much more effectively. Projects will

start to come in on time, within budget and with predictable reliability.

**CAI: In 2001, a survey conducted by QSM showed a decline in software organizational productivity by 40%. What influences precipitated this decline in productivity?**

**LARRY PUTNAM:** There was no one single cause. However, I can think of two primary factors.

First, Y2K dominated the development activities for several years before Year 2000. Lots of development activity was deferred while all of these systems were made Y2K compliant or hastily replaced by new systems to meet the deadline.

Second, the advent of web-based development came about with a host of new languages and methods to support this development. And any time a new development paradigm emerges, the old support and development tools become obsolete. We have to stop and develop new tools that let us develop and test in that new environment. This takes time and effort and lowers productivity. Paradigm shifts lower productivity while we learn to assimilate new technology.

**CAI: Where is software productivity today and what factors are currently influencing it, for better or for worse?**

**LARRY PUTNAM:** We seem to be on a productivity plateau now. We recovered from the drop around the year 2000. We gained back much of what we lost then, but we now seem to be stagnating because of the growth in expectations – a growth that is not being accompanied by a commensurate growth in productivity enhancing tools or processes.

Systems again are very big now. Organizations want lots of functionality. They want everything automated and they want very sophisticated applications and solutions. We're not just automating back office accounting operations (which used to be done by herds of little old ladies in tennis shoes). That's all been done. Now we are building

third generation systems- new things that never had any manual counterpart before. Design is hard if you haven't done anything like it before. It takes more time and effort for what appears to be a similar amount of function, but the function is much more sophisticated and difficult. Consequently, your productivity will be lower. One would expect it to be lower.

Just look at the UAD drones that we have flying in the Iraq and Afghanistan war zones. They are being controlled in real-time by men sitting in front of consoles in Ellis Air Force Base in Nevada. They communicate by satellite halfway around the world to an unmanned aircraft. They take pictures which they beam to troops in the field in both Iraq and Afghanistan. Or they release weapons and guide them precisely to targets - all in real-time, with great precision. It's all software driven. Software we never had before. Of course, that type of software is very difficult to build. But that is what's needed today. It's what's needed and it's what's being produced. But it's not being produced with high productivity because it's the first of a kind and that's always going to be very difficult work.

**CAI: In your opening remarks you mentioned that QSM was focused on process improvement. What do people really mean when they speak of process improvement? How can process improvement play a role in improving software productivity and reducing the budget overruns and schedule overruns that have historically plagued our industry?**

**LARRY PUTNAM:** Process improvement means taking action to improve the tools and methods that are used by software developers so that they can become more efficient at doing their jobs. This results in the ability to create the same amount of function in less time with less effort while obtaining a higher reliability product. That's what I think process improvement means.

If a company embarks on a real dedicated process improvement program, they will become more efficient. They will produce the same amount of function faster and for less cost. They will create fewer errors in the process and that will mean that their reliability will be increasing. By doing all of this, and by measuring it along the way,

they will be able to quantify the improvement and show the results in dollars and cents. In other words – *they will be able to show the business case*. Moreover, if they can couple the measurement process with a good estimating system, they will be able to estimate their future work more reliably. This, in turn, will further reduce the overrun and slippage problems.

Showing the business case, though, is key. Management must become convinced that measurement, estimating, and process improvement all have distinct quantifiable payoffs. That shouldn't be difficult. The fact is, the payoff from software process improvement initiatives is frequently high enough to pay for all of the process improvement tools while still providing a monetary return over and above such outlays.

**CAI: What is the importance of software measurement in a process improvement program?**

**LARRY PUTNAM:** Professor Richard Noland from the Harvard Business School once said, "If you can't measure it, you can't manage it." Tom DeMarco has made similar remarks. If you don't measure a process improvement program, how are you going to know if it's working or not? Watts Humphreys has said, "If you don't know where you are going, any road will do."

Measurement at the start of your process improvement program establishes your baseline. This is your stake in the ground. At a later point, the measurement of size, time, effort, cost, and reliability will tell you how much you've saved and what your ultimate productivity improvements amount to.

QSM uses a metric we've developed called a process improvement parameter. In our tools we refer to it as a productivity index rather than a mathematical parameter. The neat thing about this metric is that it embraces size, functionality, time and effort. The traditional calculation of productivity - size divided by effort - leaves out the time factor. However, software development is extraordinarily sensitive to schedule time. If that factor is left out of the calculation, the measurement is meaningless. That's another reason that so many products have gone astray over the years. They've been using an inappropriate and misleading metric - size divided by effort - as the basis for

making their estimates. Invariably, they turn out wrong.

**CAI: You wrote a very well-known and highly regarded book, along with Ware Myers, called "Five Core Metrics." What inspired you to write this book? What were you trying to accomplish with it?**

**LARRY PUTNAM:** What we were trying to do was to demonstrate to software managers and business executives that there were metrics that could guide them in their software development projects and that they needed to be used in a comprehensive measurement, estimating and control system.

You will often find that the actual history of a project will deviate from the plan once you're underway. Now if the deviations are too great and they continue to diverge, this is a call to action. You've got to revise the plan, usually on the fly. This is the domain of our other estimating tool, SLIM Control. It measures these deviations. It indicates when things are out of control, and it allows us to do an adaptive uptake to suggest a new plan that puts forth a new schedule and a new staffing plan that is consistent with what has happened so far.

This tool gives remarkably close estimates of real finish times from as early as one quarter to one third of the way into the project. Some companies don't like what it tells them; they don't want to hear the bad news. Traditionally, if people see bad news, if they miss a milestone, they will say, "Well, don't worry sir, it's okay, we'll make it up by the next milestone." And then the next milestone comes and they don't meet it and they just repeat this process until two weeks after they're supposed to deliver the final product and then they say, "Oh, sorry sir, but it's going to be eight more months." That's been traditional history. In response to this, our tool helps gather and analyze accurate data points so that organizations can hone in on real endpoints.

When it's all over we end up with some very good, very reliable data. We take that data and feed it into a history repository and that then becomes the domain of our measurement benchmarking product, SLIM Metrics. With SLIM Metrics, we've got 20 years of data and over 7000 completed projects at our disposal. Moreover, SLIM

Metrics is adaptable to any organization's development environment. It quickly becomes useful and very helpful to companies with only a modest amount of their own data. They don't need 7000 systems. They can lean on us for the 7000, put in 10 of their own, and they will be in very good shape to move forward from there.

**CAI: In your book you talk about core metrics. What are the five core metrics and how do they relate to each other?**

**LARRY PUTNAM:** The 5 core metrics are size, schedule, effort, reliability, and the process productivity parameter. They are all related to each other. Size, time, effort and process productivity are all related in terms of "the software equation."

Conceptually, the "software equation" is "size equals process productivity multiplied by effort multiplied by time." Now I have a little bullet list here, where the first bullet is:

- Size is the amount of functionality in source lines of code (or in any other valid measure).

Size doesn't have to be lines of code. It could be any other valid measure such as function points, use cases, function units, number of requirements, etc. Size is simply the number of countable entities that make up the quantity of function that has to be created in the development project.

Now the remaining bullets:

- Time is the elapsed calendar time of the development schedule. Time is measured in time units: days, weeks, months, years.
- Effort is the amount of human work that has to be done to build the software project. Effort is the sum of the man hours applied over the time period of the project and this is measured in person weeks and person months.
- The process productivity parameter is the metric that measures how efficient a software development process is. The process productivity parameter is a derived metric based on the time, effort, and size of the project, and we calculate it using that software equation in reverse; i.e. on a completed

project we know the size, time, and effort so we use this to calculate the process productivity parameter.

Although reliability is not explicitly stated in the equation, it is implicitly related through size and effort, so it does change with the other parameters. For every set of size, time, effort, and process productivity parameters, there is also a unique reliability parameter.

This is all related. Change one of the parameters and the others will also change. It is a dynamic, interrelated set. This is what makes this estimating system so unique among other available estimating systems. It has a rich array of trade-off solutions that are possible. If one solution is too expensive, you can lower the man power, which reduces the effort and causes elongation of the schedule. It also improves your reliability because fewer people will be working on the project, fewer defects will be created, and thus fewer defects will have to be found and fixed before it's good enough to use in commerce.

**CAI: You've mentioned SLIM throughout the interview. What are some other tools and how are they differentiated from each other on the market today?**

**LARRY PUTNAM:** Estimation tools provide a way to systematize the measurement and estimating process. They give consistency to the process and make the decision making process more predictable.

Tools tend to be differentiated by the amount of functions they provide and the underlying algorithms that each tool supplier uses. Beginners often start off with inexpensive, basic estimating tools. If they're serious about estimating they soon find that the basic tools don't have enough function to meet the increasing sophistication that the user realizes they need, especially if they start getting involved in what-if trade off analyses and alternative scenarios that bosses frequently ask for. But it's a great way to get started: you start with the basic system. And it's certainly better than the guessing game.

Sophisticated developers find they need tools that can deal with many different

environments in a wide array of trade-off scenarios. They need tools that can be easily calibrated to fit their own people and processes. They need tools that are sophisticated enough to work, yet simple enough to run and use. More sophisticated tools are usually licensed on an annual basis, because the tools are usually being updated continuously, and the customer needs ongoing support. Training is typically required for users to become proficient in the use of such tools.

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