



Presents
An IT Metrics and Productivity Journal Special Edition

Focus on Dr. Christof Ebert
A CAI State of the Practice Interview
December, 2006

Biography of Dr. Christof Ebert

Dr. Christof Ebert is Director of Software Coordination and Process Improvement for Alcatel. His current responsibilities include establishing shared software platforms and leading Alcatel's global CMM/CMMI programs. A senior member of IEEE, Dr. Ebert lectures at the University of Stuttgart and serves as a keynote speaker and on program committees of various software engineering conferences. Since the end of the 1980s, he has been an educator, researcher and consultant in software measurement. He is a member of the editorial board of the Journal of Systems and Software and is IEEE Software associate editor-in-chief. He serves on the board of the German Interest Group on software metrics within the German Informatics Society (GI). He is also the co-author of *Best Practices in Software Measurement*. Our interview between Christof Ebert and Michael Milutis, Executive Director of the IT Metrics and Productivity Institute, took place in June of 2006.

◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆

CAI: Could you tell us a little about yourself, your background, and what you're currently working on today?

CHRISTOF EBERT: I am currently Director for Performance Improvement at Alcatel, where I report to the Chief Technology Officer. Alcatel designs, develops and builds innovative, competitive communications networks. The purpose of my work is to ensure that we have the best possible R&D processes and tools in place in order to continually improve and fulfill our commitments to the different business divisions of Alcatel.

I studied electrical engineering and software engineering at Kansas State University and also at the University of Stuttgart in Germany. After earning my PhD in software engineering I joined Alcatel, where I soon became owner of the software measurement process. I have since worked in a variety of different areas including engineering quality. For the past five years I worked with the CTO on a number of different initiatives before taking on the directorship of our R&D performance improvement efforts.

CAI: The term “productivity” gets used quite a lot in our industry. Nevertheless, it seems that many people have different assumptions about what the term really means. How would you define software productivity?

CHRISTOF EBERT: The precise definition of productivity is output divided by input. That means that when you create something based upon fixed resources, the output is directly dependent upon those resources. So if one chicken produces one egg per day and another produces two eggs per day then – assuming that the chickens have both eaten the same amount of food - the second one would, by definition, be more productive.

However, in the more complex world of software and systems development, productivity is not only a function of output and input, it is also a function of the environment. In this case, environment might refer to people or location or even the nature of a project. That means that if a programmer produces a certain amount of code per year while another programmer, in a different environment, produces much less code, that second programmer might still be more productive than the first. That’s because the second programmer might be working on a much more challenging task with more requirements while the first may simply be doing very routine work.

In software, we have traditionally measured productivity by taking the produced code as an output and then normalizing that with the inputted effort. This “lines of code” (LOC) approach to productivity still dominates today for the simple reason that it’s easy to reproduce and simple to measure. However, the LOC method is too simplistic when it comes to making comparisons across different companies or even across different

product lines. Metrics such as function points, by comparison, are able to evaluate output not only in terms of delivered functionality but also in terms of environmental parameters.

CAI: How would you describe the state of software productivity today? What do you think we need to do over the next 5-10 years to see significant improvements?

CHRISTOF EBERT: Generally speaking we have seen tremendous progress in software productivity over the past few decades. What you can buy for a buck today represents so much more than what you could buy - with that same buck - 5 to 10 to 20 years ago. This has been true for any type of software application.

But it's not just about cost. In addition to being cheaper, the applications today have much more functionality. Some might say, in response to this, that not all of what you buy today has real value (i.e., that sometimes you are paying for functionality that is there but that is not really valuable to you). But if you put that aside, there are clearly a lot of things which were traditionally not a part of applications or embedded systems but which today have become standard and even expected.

In other words, in the final analysis, you are getting more today for the same amount of money. This trend has been very clear. You can even see it in some of the metrics that I spoke about in your previous question. Specifically, the delivery of function points or software code person years has increased dramatically over the past two decades.

A lot has been done. However engineers often fall into the trap of explaining productivity in terms of technicality rather than value, especially when challenged by customers or senior managers to explain why software development costs so much or why it is not possible to get it done cheaper and faster.

CAI: Some people argue that software productivity improvement is simply not possible on account of inherent constraints or natural laws. What are they

referring to? Do you agree or disagree with them?

CHRISTOF EBERT: We have to acknowledge that the complexity of software is growing at a faster rate - in terms of our capacity and competencies - than we can cope with. From that perspective, we are doomed to experience reduced levels of productivity. Some have gone so far as to call this particular constraint a "natural law."

However, in my opinion, this is not the most important issue. Coming back to my earlier statements, I believe that we need to evaluate projects more from a value perspective. In the words of Fred Brooks, in his famous article "No Silver Bullet: Essence and Accidents of Software Engineering," complexities are either necessary or "accidental." He went on to state that there are both essential and accidental elements which contribute to the difficulty of creating a product. What he calls "essential" is what I am attributing to "value." If I have to deliver a system within a specified time frame with a certain degree of quality following a certain set of fundamental requirements, then this is essential. This is something I cannot escape. However, we often create accidental difficulties on top of these essential requirements and it is these accidental complexities that nobody expects and that nobody is really paying for.

We can easily improve productivity - and this was acknowledged by Fred Brooks in the 70s - by focusing on these so-called accidental complexities. We can remove frictions along our code, remove redundancies, reuse what we have, make sure we have good tools in place and automate certain workflows. These are all things that address accidental complexity and that can also help us overcome the strange belief that it is somehow natural to be stagnating, from a productivity perspective.

From my point of view we are not stagnating. We can see that in the software which is being delivered and in the amount of value which is being generated by software. The world would fall apart without software. Moreover, our world today is governed by a lot of embedded systems. In fact, the vast majority of software today is going into embedded systems. Being embedded means that these systems are hidden and that they escape our awareness. Nevertheless, such systems are all pervasive and they keep the world running.

CAI: You've mentioned the GQM method in some of your papers. What exactly is the GQM method, and how can it be used by organizations to improve their overall productivity?

CHRISTOF EBERT: Goal-Question-Metrics, or GQM, was created in the 80s and 90s in order to focus measurement programs on what they were trying to achieve. GQM states that the ultimate objective of any metrics program should be to achieve a specific business goal. Using the methodology, you are guided through a series of questions that revolve around the changes and behaviors that are going to help you achieve your goal. Only after asking these questions do you evaluate the metrics. You then determine how to measure your progress towards the stated goals.

Goal-oriented measurement helps ensure that what you measure is useful with respect to the goals you want to achieve. To provide a concrete example, when assessing productivity our first question should not be "What is our productivity?" We should rather ask, "Is my productivity this year better than last year's productivity, for the same system?"

CAI: You co-wrote a book with Manfred Bundschuh, Reiner Dumke and Andreas Schmietendorf: *Best Practices in Software Measurement*. What inspired you originally to write this? What did you set out to achieve with it?

CHRISTOF EBERT: At the time we decided to write this book, there was already a lot of good literature available on the subject of software measurement. However, we wanted to create a book that combined the usage of measurement with a lot of practical experience from industry. To this end, we pulled together four authors who represented very different industries and backgrounds. One of us comes from a university environment, one from telecommunications, another from a bank and yet a fourth from the insurance industry.

Our mission was to address measurement systematically and to create a book that combined the basics of software measurement with how-to guidelines that would allow practitioners to directly transfer the book knowledge to their own work lives. While investigating such topics as estimation, quality improvement, productivity

improvement, supplier management, and information systems performance we always set out to include concrete examples that users would be able to practically relate to their own work.

The feedback in the two years since this book was published has been very good. In fact, we are currently in the process of producing a second edition which will probably appear sometime within the next year. We will look at what has changed over the past 2-3 years (admittedly, a lot) as well as what we've learned since then.

CAI: Why is the measurement of productivity so difficult? What are some of the most common pitfalls and challenges that people encounter when they first attempt to go down this path? What advice can you offer?

CHRISTOF EBERT: At the beginning of our conversation I stated that the definition of productivity was output over input. I went on to further define productivity as a function of output, input, and environment. Of those three components, I would say that the only simple element is input. It is easy to determine how much effort was put into a project. However, the input figure is not fully complete in itself, since you still have to consider whether or not the effort was completed under time pressure or not. It is at this point that you must introduce the element of environment.

The component of this equation that is most difficult to obtain is output, or value. Value is ultimately determined by the market and by users. When you examine this, you must somehow differentiate between perceived value, which is what people are paying for, and the internal or system-based value. I strongly recommend taking a pure systems perspective towards value which, in turn, means breaking the system down into components. At that point, you can relate what you are working on to what the user is paying for. This may sound trivial to those of us who are experienced in requirements, but it is not always so easy for the engineer.

It is a very good approach to always relate each of your tasks to the end value. If you are able to have value hierarchically broken down like this, into the very fabric of your day to day work, then you should be able to easily produce traceability matrices and strong business cases and this, in turn, will enable you to talk about output in a clear

and consistent way.

A common pitfall I see in productivity measurement is when organizations frequently take a standard formula - such as lines of code per person year - and then let that metric stagnate without revisiting the calculation to see if it still makes sense.

Something that happened at Alcatel, for instance, was that all of the process improvement tools we were using on a specific product line - CMM, agile methodology, etc. - revealed that our perceived productivity was remaining stable across each of the annual releases of the product. Senior management looked at the unchanging productivity figures and said, "Evidently your improvement programs don't give us added value." We then looked at our data more closely and determined that each of the annual releases of the specific product line had much more complexity and content architecture than the previous release. Our project complexity was increasing which meant that the value we were providing was greater. We were creating more value with the same effort. However, by using only a one dimensional productivity metric - and not taking environmental parameters into account - we had fallen into a typical trap of letting the metrics stagnate.

CAI: Could you elaborate on some of the most common productivity metrics that are in use today as well as what differentiates them from each other?

CHRISTOF EBERT: The single most popular productivity metric is the lines of code measurement (LOC). LOC measures output by examining how much code is new or changed and then compares that figure to the amount of effort invested in the project. The reason this method is so popular is very simple: you can create the measurement calculation from your final product and not have to look at raw data. The figure can be easily normalized and standardized. It is fairly easy to generate and can be automated to a good degree.

Function points are more complex but they approximate the true value of a project much better than lines of code. LOC has limits and - without taking environment into account - does not necessarily reflect true productivity. The numbers could turn out differently based on a variety of different working styles. For instance, I can imagine

for a given problem statement that a programmer could write a lot of code or a little bit of code and potentially solve the problem either way. Likewise, a programmer could re-use code or generate it anew, which is something that will also effect the LOC measurement. Function points overcome these limitations by measuring the eventual complexity or difficulty of the product and by also incorporating environmental parameters into the calculation, parameters such as programming environment, skill level, amount of development science, etc.

A third category of productivity measurement that is in increasing use today is that of benchmark metrics. Benchmark metrics test the height of output over input in a single number format. When tracked over time, this dimensionless parameter helps you see how you are progressing within a given industry. These measures take output, input, and environmental parameters into account but in the end they produce a single number which is something that is much easier to track and make use of over time. With such a metric, you would be able to see, for instance, that year after year you were achieving an improvement of x percent. You would then be able to ask questions such as, "Do I have deviations for certain product lines? What is the explanation for these deviations? Do I have some products which have higher productivity than others? What could I learn from these discrepancies?"

CAI: What kind of questions should organizations ask themselves from an analytical perspective once they've identified the right metrics and put good collection mechanisms in place? How are they to figure out how to interpret their data? And what are they to do with it after they've interpreted it?

CHRISTOF EBERT: First, we must find out whether or not we are better or worse off than our competitors within the various different product lines of our portfolio. Second, we must find out whether or not we are improving over a certain time frame. These are the two most important questions. The first question measures performance versus the market. The second tracks performance across certain time periods so that you can see whether or not you are moving in the right direction.

These are questions that senior managers should be asking themselves every morning:

Are we in the right market? Are we moving forward? Are our efforts good enough? Such questions are not necessarily software specific; they would be necessary for the evaluation of any portfolio.

I find the second part of your question interesting. Coming back to what I said a bit earlier, you should be continually asking all of these questions primarily to set goals. You should first determine where the enterprise or product line has to go, and then relate that to your current status. From there, you can set targets for next year or next quarter. To measure this you can create dashboard metrics using standard measures - with respect to customer needs - such as speed, quality, accuracy, and quality. These will help you determine whether or not you are still on course.

Once you've integrated these dashboard metrics, they can become a very effective management tool. The aggregation can be useful for portfolio management, better scorecards, benchmarking, etc. You can also make use of such a tool for the evaluation of strategic decisions such as which products to eliminate or what kinds of acquisitions to embrace.

CAI: How does an organization get started in process improvement after having heard all of this?

CHRISTOF EBERT: From a practical perspective the very first thing to do is to find out what really matters. You need to know the value proposition of your project and understand exactly what people are paying for. You could easily improve a lot of things without this analysis. However, if you don't fully understand the project's value, you might find out that nobody is interested in the improvements.

The second step is to determine the customer's specific goals, goals related to needs and challenges of the customer, cost sizing, targets, etc. You then need to ask how best to achieve these goals.

Third, you must ask what measurements will best help you determine if you are progressing satisfactorily towards these goals. I referred earlier to the SEI core metrics such as size, effort, time, and quality. Depending on what your objectives are, you

might want to add cost or cost breakdown to this list. You should also consider balancing these project metrics with some product or business oriented metrics, because – in the end - we don't work for the sake of creating a lot of software; we work for the sake of having a growing business.

At this point, you can take five of these measurements (no more than eight) and set up a baseline of your current performance. With such a baseline in place you can track progress and identify ways to implement improvement initiatives to help you further achieve your objectives.

Questions? Suggestions? Comments? Please contact the IT Metrics and Productivity Journal Editor at michael_milutis@compaid.com