

End Your Frustration With Software Development

Lawrence H. Putnam and Ware Myers

The constant litany we hear from high-level managers above software development projects is that projects take too long and cost too much. Then they complain that they don't find out that projects are in trouble until it is too late to do anything about them. And finally they ask, "Am I getting good value for the investments I am authorizing for process improvement?" In fact, they wonder how they can tell if their project people are working more effectively this year than they did last year.

Well, you can't tell much just by looking. That is the nature of knowledge work. But still there is *work* going on. It is not all fun and games! How have managers of earlier forms of project work solved this problem?

Take building construction. The client wants to know the cost and the schedule; that involves estimating and bidding. The construction manager wants to know that construction is coming along as estimated and bid; that is project control. Construction executives want to know that their building process is getting better; that is productivity improvement.

You know the answer, of course. The construction people had to measure some things and then turn those measurements into what we will call *information*. Information is measurements that lead to action. Software people construct something, too, a program; that work is analogous to what construction people build.

Unfortunately software is not bricks and mortar. The trick here is to find something you can count that leads to the information that underlies estimating, bidding, control, process improvement, and defect reduction. Those somethings have been found. They are the Software Engineering Institute's four core metrics:

Cycle Time (of a project) or schedule time, such as number of months

Effort (person-months); in software development, the largest element in project cost

Size of the planned product, in a unit like lines of code, or more broadly, the amount of functionality

Defects, as a stand-in for reliability and quality

In addition, we add a fifth metric: *Process Productivity*, derived best from Time, Effort, and Size of past projects, where these values are now known quantities. These are the five core metrics we need to answer the questions that managers have.

Before we look into the relationship between the core metrics, we note that employees and often managers, too, fear measurement. "How do I stack up?" they ask. "Will the results be used against me? Will they impact a promotion? Will I get laid off or fired?"

The answer is that metrics used for estimating, bidding, planning, control, process improvement, and defect reduction must be used only for those purposes, never for personnel actions. Employees and managers must have confidence that those are to be their only uses. In fact, when metrics are effectively employed to these ends, employees and managers become more secure because software development will be running more smoothly.

The Relation Between The Core Metrics

In order to make use of the core metrics for project purposes, we must first understand the relation between them. First, let us consider this relationship on any kind of project, whether it is construction or software development. It is something along these lines:

The amount of functionality (Size) developed (a building or a software product) is the product of Effort applied over the Time period of the project at a level of Productivity.

In the case of construction, of course, the relation becomes more complex because you have the cost of building materials and construction machinery to add in. In the case of software development, however, the relationship is largely confined to these four metrics; the cost of desk space, computers, and other overhead over the period of a project is known.

Our first step was to find what the actual relation between these four variables is. To do that we diagrammed two relations: the one between Schedule Time and Size, and the one between Effort and Size.

The behavior of Time and Effort and Defects is different; both increase with size, but schedule increases more rapidly with smaller size projects and flattens out with large projects. Effort and Defects increase slowly for the smaller size projects but increase rapidly with large size projects. Figure 1 illustrates this behavior.

values is, of course, more accurate than obtaining it in some way less dependent on metrics, such as from an “expert” group who sit around for a session or two chatting about it.

One more catch: the company has to have a “process” that it can count on repeating, or at least come close to repeating. Organizations in Level One of the Software Engineering Institute’s Capability Maturity Model, by definition, fail to have a repeatable process. So, climb up to Level Two, at least in that respect.

Obtaining a value for Size is not as easy as we may have made it sound when we called it “known.” Early in the game, when some high executive claps his hand to his forehead and exclaims enthusiastically, “Let’s automate our secretaries,” we don’t have the least idea what the Size of that software is going to be. As we collect requirements, subject them to analysis, see if we can find an architecture, smooth out the big bumps (commonly known as risks), we gradually refine a value of Size that we can use in the software equation.. It is not precise; that doesn’t come until the end of the project. But it is good enough to use in estimating.

At any rate, at the point of estimating we have reasonable values of Process Productivity and Size. Those values enables us to draw the straight line sloping from northwest to southeast of a log log diagram of Time and Effort, as shown in Figure 4. Somewhere on that line is the particular combination of Time and Effort at which we have to operate. (Actually, because of the uncertainty of the Process Productivity and Size values, the sloping line is at the center of a region extending a bit to the right and left of the line; that is, uncertainty exists but we can ignore it at this point in the discussion.)

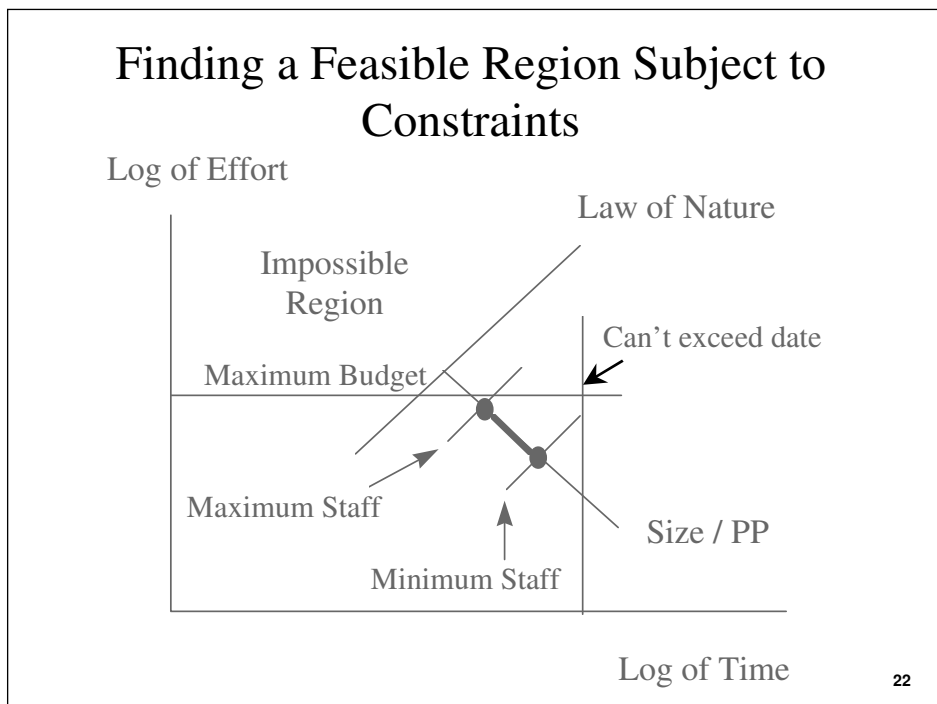


Figure 4. The line pointing toward the southeast represents the values that Time and Effort can take.

First off, we can cut off the region northwest of our line. That is the “Impossible Region.” It is impossible because no project of this Size at this level of Process Productivity has ever been accomplished in that short a Time, no matter how much Effort has been applied to it. This “Law of Nature” derives, indirectly to be sure, from another law of nature: nine women cannot produce a baby in one month.

Second, in the practical world there are real-life constraints, shown on the figure, that limit our estimate to a small portion of the line, designated by the small circles. Anything outside this small region is not feasible to try to do.

Using the Core Metrics to Control

The employment of the core metrics does not end with a successful bid. The next task is to find out whether the team is accomplishing the project on the Time schedule and within the Effort on which the bid was based. To implement this continuing control of the project, the project manager has to spread the person-months of Effort over the Time period agreed to with the client. Then, he/she has to compare the person-months actually being worked each month with the planned schedule. All too often staff is not available in the quantities and composition contemplated by the plan. Obviously, trouble looms.

Still, what really counts is whether the project is accomplishing the work as planned. For bidding purposes, we had measured the work in some unit of Size, such as source lines of code or function points. For control purposes we have to compare the number of units of Size being accomplished against the number planned to be completed week-by-week or month-by-month as the project progresses. Also we can use other units pertaining to various aspects of the work accomplished, such as those illustrated on Figure 5.

A Dashboard Makes it Easier to Get the Real Picture

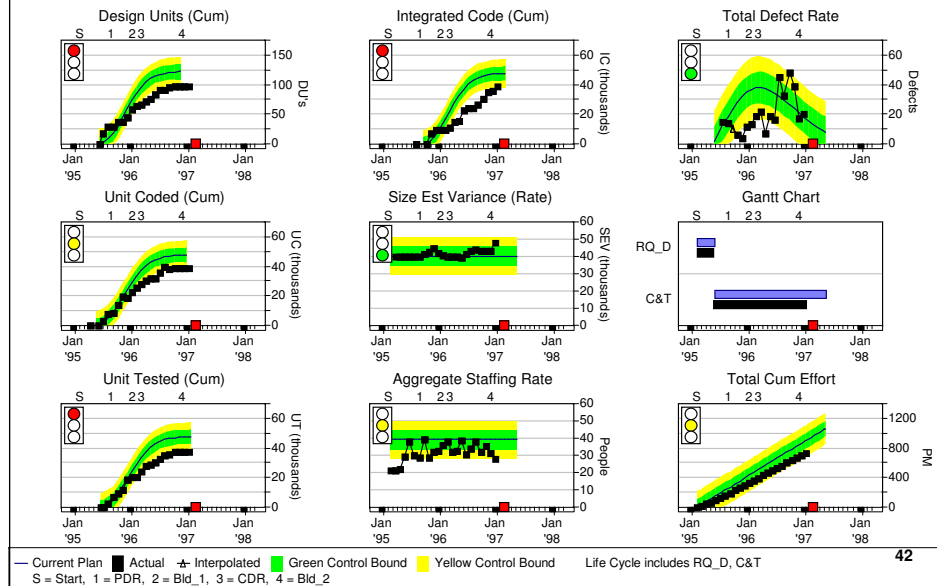


Figure 5. A project can project the work it is to accomplish in various units of measurement in addition to Size and check periodically to see that the work accomplished matches the plan.

Studies of project execution find that approximately one third go as planned. They are a living proof that it is possible at the current state of the art, if the art is artfully employed, to develop software as planned.

Less happily, these studies find that another third do not go as planned and bid. The functionality (Size) needed turns out to be greater than that planned, requiring more Time and Effort (Cost). Or, the project runs into complications (risks) not anticipated. Or, the project manager was unable to staff the project as planned and has to have more Time. Or, the functionality originally planned cannot be completed within the Time and Effort bid, but the client decides that the functionality completed is adequate for current operations. There are a great variety of these combinations, but something usable eventually goes into operation.

Obviously, each of these revised outcomes is a subject for negotiation between the project and the client, sometimes with the assistance of lawyers. All of the outcomes involve either less functionality (Size) or more Time and Effort. Inevitably, these less-than-planned outcomes impact the business operations of the client. Unhappily, they impact the emotional state of all involved with deleterious effects on the project itself. Software people have been known to leave the profession.

Finally, these studies report that the last third of projects are not completed at all. That is, the client does not get the benefit of more functional software. In a sense, the project lives

on, at first in the hands of haggling managers, then in the professional hands of lawyers, and sometimes finally in the lofty precincts of a court. No software is coming out, but costs continue as if it were. Effective use of the core metrics prevents this unhappy outcome.

As we observed at the beginning of this article, one of the frustrations experienced by executives necessarily focused on commercial issues is that they always “find out that projects are in trouble when it is too late to do anything about it.” In the absence of running control methods, that is bound to happen. With an assessment of the state of each project weekly or monthly, signs of trouble become evident in time to correct the cause of the trouble, or adapt gracefully to it. For instance, if the project is not building up staff as planned (an early indicator), trouble lurks down the trail a bit. If the project is not turning out units of accomplishment as planned, something is awry, either the plan or the achievement. It bears looking into, now rather than when the lawyers eventually arrive when it is too late to do anything worthwhile about it.

Using The Core Metrics To Improve

Process Productivity is derived from the other three core metrics. It measures not only the amount of output per unit of Effort, but also the impact of the Time schedule taken by a project. Therefore, it potentially answers the other two frustrations that commercially minded executives express with respect to software development:

Projects take too long (Time) and cost too much (Effort).
Is the investment in process improvement paying off?

First, let us see just what Process Productivity is:

Process Productivity = Size divided by (Effort x Time)

It is evident that as either Effort or Time (taken to complete past projects successfully) increases, Process Productivity decreases. In reverse, as either Effort or Time taken decreases, Process Productivity increases. That makes sense. Getting a job done faster with fewer people is the essence of productivity in a process relationship.

The addition of Time to the Process Productivity relation, as compared to the absence of Time in the traditional or factory version of productivity, thus, has a significant effect on the values expressed. In other words, to the extent that management can take steps to improve Process Productivity, it can alleviate the worries expressed in its first complaint: Projects take too long and cost too much.

As to the second complaint, Is the investment in process improvement paying off; there are two approaches to a favorable answer. The first might be called non-numeric. In one application of this approach, the Capability Maturity Model, management does its best to implement the several hundred good practices recommended by this methodology. Then, every few years it calls in an evaluation team to study the degree to which some of these

practices are actually in use. The team, as a matter of judgment, ranks the organization in one of five levels.

In another non-numeric approach the desirable practices are set forth in a specification to which management directs its software projects to comply (and hopes that they do).

The second approach is numeric. Management obtains the Process Productivity number for each completed project. If the number is increasing with each successive project, that project organization is getting better over time.

Of course, there are matters for judgment even in this numeric approach. For instance, there cannot be so much change in the composition of the team as to affect its productivity substantially. The application type needs to be the same. For instance, real-time software is much more difficult to produce than business software. Ordinarily, team composition changes slowly and business software teams stay with business. Under circumstances such as these, the Process Productivity number is a good numeric gauge.

Process Productivity is derived from only three metrics, but they are *core* metrics. The result actually reflects a broad range of capabilities: specifications, training, techniques, skills, tools, methods, users, management—just about everything about each project. It answers management’s second concern impartially: Is the investment in process improvement paying off?

Figure 6 summarizes the current status of Process Productivity in nine categories of software development, ranging from business systems to avionic and microcode development.

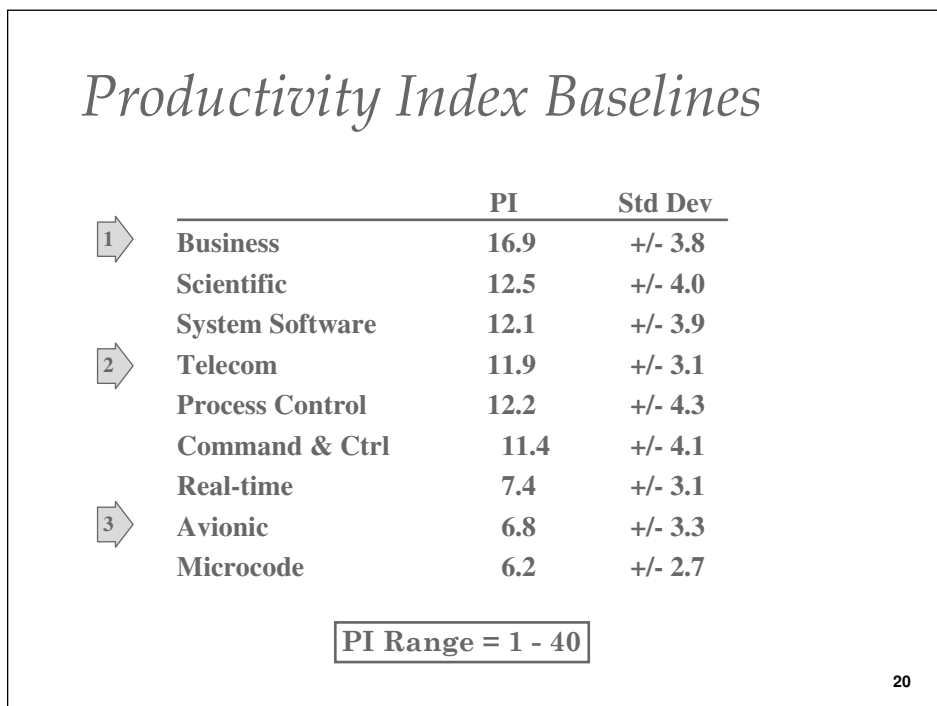


Figure 6. A few organizations have achieved Process Productivity index values in the 20's, showing what is possible at the current state of the art. For the rest, there is plenty of room for improvement.

Using the Core Metrics To Reduce Defects

The first four core metrics—Size, Time, Effort, and Process Productivity—are not chiseled in stone. To a considerable extent they are under the control of the two managements involved in a project, the client and the developer. At the time a project is under discussion, Size, Time, and Effort can be modified by agreement of the two managements. The fourth metric, Process Productivity, can be improved by the developer-management side over a period of time by carrying out an effective process improvement program. Appropriate improvement of one of these metrics also has the benefit of reducing the Defects.

Size. The number of Defects increases with the Size of a project, as Figure 7 shows. This increase is not surprising; the more work, the more errors committed. However, it is often within the authority of the two managements concerned to limit the Size of the project under consideration. For instance, it may be divided into an initial build and subsequent builds, each one smaller than the entire development would be. Only the most urgent needs are met in the initial build.

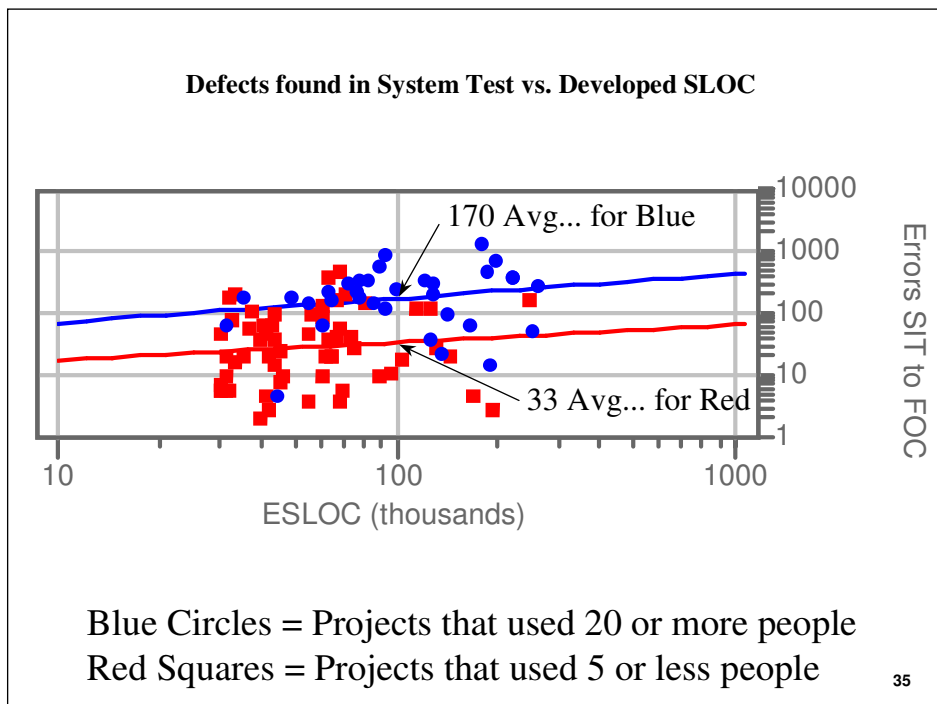


Figure 7. Defects found between System Integration Test and First Operational Capability increase with the Size of the software product. In addition, similar size

projects employing 20 or more people committed substantially more errors than projects with five or fewer people.

Time. It is a matter of common sense to conclude that, if a project is hurried, the developers commit more errors and later find more Defects. A project is indeed “hurried” if it is scheduled to be completed in the minimum development time, just in excess of the Time values in the Impossible Region. Development at that point in Time is possible, but at the cost of increased Defects. In general, the allowance of more schedule Time, up to about 130 percent of the minimum development time reduces the Defect rate and reduces the effort and associated cost.

Effort. Similarly, if the managements can reduce the Effort planned, that reduction will also reduce the errors committed. Other things being equal, fewer people commit fewer errors. This reduces Effort but it does take longer. Lengthening Time, reducing Size, or improving Process Productivity all permit reduction of Effort.

Process Productivity. Improving this factor also reduces the number of Defects.

Thus, judicious selection of Size, Time, and Effort for a project will result in a reduction of Defects. Success in improving Process Productivity has the same effect.

No Simple Solutions But...Intelligent Choices

Industry leaders recognize that measurement underlies the effective execution of business and industrial processes. They realize, however, that ham-handed measurement leads to trouble with the people being measured. They approach measurement with these principles in mind:

Measure the process, not the people.

Start with a minimum data set—the five core metrics.

Set realistic goals, based on where the organization then is, usually something short of superb!

Identify strong points and build on them.

Identify bottlenecks and fix them.

Take action to improve the process.

Underlying the intelligent management of software development then is the judicious employment of just five core metrics. They are the basis for laying out the development plan in the first place, a plan that is realistic and doable. They are then the basis for control, that is, seeing to it that the project is following this realistic plan. Or, sometimes if it is swerving off plan, that the metrics provide the basis for correction.

Then management faces the longer cycle: Improving the productivity of the software development process is no simple matter; as we said, there are no simple solutions. The five core metrics do give us the basis for intelligent choices.

Finally, treat measurement as an ally, not as an adversary. Everyone is better off in the cooperative, effective organization.

Note: For more detailed information, see our 2003 book, *Five Core Metrics: The Intelligence Behind Successful Software Management*, Dorset House Publishing, New York, 370 pp., and our Website: www.qsm.com.