

Real Process Improvement

Steve Neuendorf
Independent Management Consultant

Process improvement: The name belies its nature. Many people see it as the way to build products better, faster, and cheaper. In truth, there are many ways to be better, faster, and cheaper at whatever you are doing, and process improvement is only one of them. However, process improvement is the most reliable because first it works on people and only then does it work on the process. First you understand how good, fast, and cheap a process is and why it is that way. You also develop an understanding of why alterations of a process or adoption, as well as altogether different processes may be better, faster, or cheaper. With this understanding, you can then implement the changes that will result in improved performance.

Everyone agrees on what process improvement is, right? It is the most discussed topic in software engineering. It has been around for such a long time; there is no need to define anything before launching into a familiar discussion, right? Wrong.

Process Improvement Defined

To paraphrase W. Edward Deming, author of *Out of the Crisis*, process improvement is not something you talk about; it is something you do. For process improvement in software engineering, when all is said and done, a lot more is said than done. What is worse, far too much of what is done is done wrong—the process is not better, and process improvement is besmirched. Let us look at process improvement and see if there is a way to effectively and consistently improve software engineering processes.

Two things need to be understood: process and improvement. It may sound silly, but the number of different definitions and their imprecision lead to a great deal of confusion and misunderstanding about process improvement.

A prerequisite for process improvement is to have a comprehensive process definition. The operating definition of process is “the definition of the way in which something is intended to be done.” Any process has an accomplishment objective. A process must describe the steps to achieve its objective and the means to do the steps. Since this definition could apply to many things, for software engineering, our process is assumed to have a formal definition so that it can be repeated.

Important Terms and Concepts

A variety of concepts must be understood before you can make the distinctions needed to initiate true process improvement.

Processes vs. Projects

Distinguish processes from projects: A process is the intended way to complete a project, whereas a project is the application of resources to that process.

Because projects are tangible, the only way you can measure a process is to measure projects and use the information to make inferences about the process. When measuring, it is important to know what can affect processes and projects:

- **Processes** are affected by *common causes of variation*, e.g., teams, tools, environment. When you work on common causes, it is called “process improvement.” Common causes should be addressed by your process.
- **Projects** can be affected by either common or *special causes of variation*, e.g., system failures, attrition, or improper schedule or budget. Working on special causes is (or should be) problem identification and correction, commonly called “fire fighting.” Special causes cannot necessarily be prevented or addressed through process changes.

You must always distinguish which type of cause is at work, because if you treat a common cause with special cause techniques and tools (or vice versa), you are “meddling” rather than solving problems. Meddling invariably causes more “fires,” and fire fighting does not improve processes.

Efficiency, Cycle Time, Quality

Three characteristics describe any process:

- **Efficiency (E)** – the relationship between resource use and accomplished results.
- **Cycle Time (T)** – the “design speed” of the process, i.e., the speed of a particular development process relative to other processes (assuming certain factors are equal).
- **Quality (Q)** – the quality of the process.

You may laugh when someone says “good, fast, cheap—pick two” but this is more insight than humor. For any process,

function $f(E,T,Q) = \text{constant } K$.

Therefore, if you want better *and* faster *and* cheaper, you will need a different process.

Sequence and Means

Sequence and means are the two elements of a process. Sequence is simply the order in which things are accomplished. There are two types of sequences:

- **Required** – an order that *must* be followed, as demonstrated through precedence, i.e., putting on a second coat of paint requires that the first coat already be applied and cured. In software development, precedence has long shown it is best to start coding after the requirements are gathered.
- **Discretionary** – the order in which it is *decided* that something will be done. Past precedence and the availability of related means influence the sequence in which discretionary processes are executed.

After sequence comes the means by which the steps or tasks will be carried out. It is understood that a project is “the application of resources to a process to produce a result.” People usually think of resources as labor, time, and money, but for understanding the process you must consider other resources such as tools, techniques, technology, and factors particular to “resources” such as the skill and experience of the team members. Therefore, the definition of means includes all aspects you should consider so that you can understand and improve processes. It is helpful to consider the means separately in categories such as management, technology, teams (or people), tools, techniques, and environment.

Capability and Capacity

For effective process improvement, an organization must consider all of its processes collectively. As defined so far, a process is a sequence and a set of means. By this definition, each organization would have a virtually infinite number of processes (possible combinations of E, T, and Q). Therefore, the additional dimensions of *capability* and *capacity* must be understood. Each possible combination of E, Q, and T defines a capability. The current ability of the organization to execute a capability defines capacity.

This is not as complicated as it may sound. For example, one of the means defined as “Red Team” is comprised of members with certain experience and knowledge. Red Team projects are done faster, better, and cheaper than “Blue” or “Green” Team projects. Red Team performance levels are a “capability.” Because the organization may have only a limited number of employees qualified to form Red Teams, there is a limited capacity associated with the Red Team capability.

Innovation and Continuous Improvement

There are two categories of improvement: *continuous improvement* and *innovation*. Continuous improvement of processes is the systematic upgrading of lower capability means to give higher

capacity at a higher capability. To use the prior example, continuous improvement of team capability would be to provide training to Blue and Green Team members so that more Red Teams can be formed.

Innovation is the introduction of new capability. Again, using the prior example, innovation would be training everyone in a new technique. Everyone, even the Red Team, would have a greater capability to execute the new technique. It is important to note that innovation usually introduces learning curve dynamics and a risk of failure to a greater extent than continuous improvement.

Following is another distinction between strategy and process improvement. Again, using our example, if you were to merely adopt a strategy to replace Green and Blue Team members with Red Team-qualified candidates, better values of performance (E, Q, and T) would be expected. However, Red Team-caliber candidates are expensive and much harder to find. Green and Blue Team members could be trained (their processes improved); however, process improvement requires understanding the process—you need to know what will improve a process and how much improvement is needed. For example, the cost of making your Blue and Green Team members perform like the Red Teams must first be determined, then the benefit of improving team performance can be discovered.

As in anything complex, what is “obvious” is not always true, and what is true is not always obvious. Only by improving your understanding of the process can you manage the risk of making changes that do not result in the desired improvement.

Tools for Process Improvement

For all of the above categorization, there are still two categories of process improvement that need to be considered when you apply the available process improvement tools. There are repetitive processes, such as most manufacturing, and there are nonrepetitive processes, such as software engineering.

Repetitive Processes

Virtually all of the common process and statistical process control (SPC) literature focuses on repetitive process principles and examples. Characteristics of these processes are a mostly fixed precedence, sequence, and means. Generally, the process flow-chart tool is used to understand and improve these processes, with care to use the decision element of the flow-chart tool to divide flow into segments that are distributed in such a manner that SPC tools can be used to analyze data (see Figure 1).

Nonrepetitive Processes

The goal in process improvement for software engineering is to improve nonrepetitive processes. Your understanding of these processes comes from analysis of the effects of variation in the means of production on performance. For nonrepetitive processes, attempts to understand and improve processes by using repetitive process tools and techniques will yield the same results as trying to teach a pig to sing—you get no singing, and it upsets the pig.

The key to understanding nonrepetitive processes is knowing that while the means may be similar from project to project, the steps and sequence can never be exactly the same. That is, even

Figure 1. *Flow charting without documenting the properties makes a nice picture but a poor tool for process understanding and improvement.*

Repetitive Process Flow Charting	
Process Flow	Symbol Charting
Symbol	Properties
Arrow	Flow volume
→	Flow timing
Input/Output	Naming
▭	Responsibility notation
Storage/Queue	Volume
▽	Capacity
	Timing
Operation	Resource use
▭	Timing
	Capacity
Decision	Criteria
◇	Flow percentage

if you start a new project with the last project's teams (people and skills), management (people, styles, and leadership), and tools and techniques in the same environment, you are nevertheless dealing with a different project. However, there are myriad aspects of the means (management, teams, tools, techniques, environment, etc.) that can be measured in a way so the variation in these measures can be related to variations in performance (E, T, and Q) for any project.

However, at this point you do not necessarily yet understand the nature of your nonrepetitive processes. Notice in the process flow-chart tool description from Figure 1 that the volume property of the arrows, along with the "normalization" use of decisions, allows an analyst to "normalize out" variation due to throughput and size. For nonrepetitive processes, project cost, defects, and duration will vary due to differences in the means of completing the project *and* the (deliberately) heretofore not mentioned size factor—changing the size of a project can change everything.

I have not mentioned the size measurement because it seems to work like a light switch—flip the switch and about half the people turn on and the rest turn off. The goal of a size measure is to remove variation due to size from the analysis without introducing an-

other source of variation and without removing a process-related source of performance variation. If your size measure does this (some straightforward statistical analysis will tell), you are on the right track.

An Example

To illustrate how this translates into reality, I use an example with which I assume most readers are familiar: the Software Engineering Institute (SEI) Capability Maturity Model (CMM). In a simplistic view, the CMM measures capability maturity on a scale of Level 1 to Level 5 with Level 5 being most mature. Several practitioners and analysts have also developed relationships between the CMM maturity level and performance in each dimension of process measurement (E, T, and Q), notably, the higher the maturity level, the better the performance. From the analyses I have seen, the differences in each E, T, and Q are substantial, and I assume they are statistically significant.

The nonrepetitive process improvement model is analogous to this view of the SEI CMM. That is, for each of the means (management, teams, tools, techniques, environment, etc.), a scale of possible conditions is developed, and criteria to evaluate that means according to that scale are prepared. For ex-

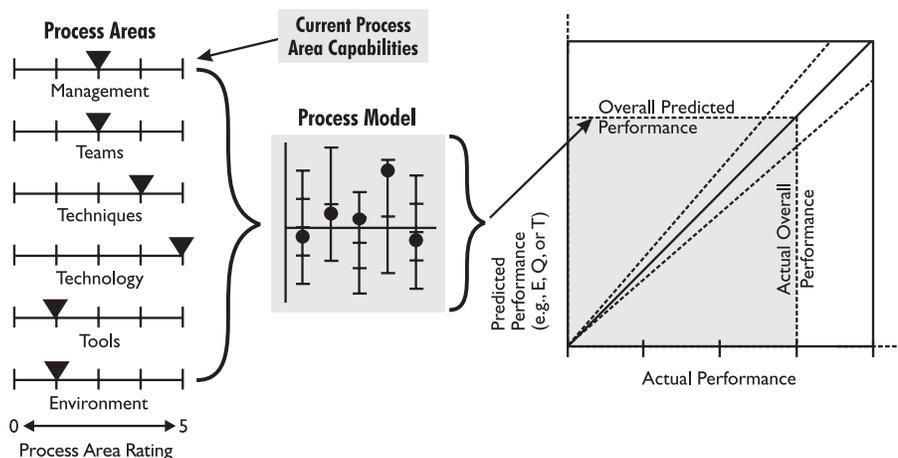
ample, imagine project management is the means being evaluated. The low end of the scale would be no project management or maybe a project lead and no formal tracking. The top of the scale would be a full-time project manager who reports to the project management office and uses a full set of project management tools.

With all means identified and a scale created for each of the means, the values of the means for any given project constitute a process. Improvement of the process is the improvement of the means. Figure 2 shows how a process is managed to obtain a performance result. For the organization measured, the scale in each means, e.g., management, represents the range of influence of the measured capability on performance. For any project, the actual value of the mean (the triangle symbol) predicts the contribution of that aspect to the overall performance of the project. The process model shows the collective effect of the measured means values on project performance and gives the overall predicted project performance. The goal of performance measurement then becomes measuring actual performance against predicted performance to detect problems. Process improvement is working on the means, both for capability (higher levels on the scale) and for capacity (how much is available for use).

If you know what to do, calibrating the process improvement model can be straightforward, i.e., for any given set of means, values for E, Q, and T (exclusive of special causes) can be predicted. It follows that if the model is calibrated, the effect of changes in the means (benefits) can also be quantified. Finally, understanding the means makes improvement actions and investments evident and makes improvement cost (time and money) easily determined. You now have the action, cost, and benefit of process improvement objectively in front of you. Now more can be done than said.

It also is important to note that even in a calibrated model, it is likely that the variation in performance effect for

Figure 2. Managing a process to obtain a performance result.



Project Management is (1) identifying the process used (and predicted performance) and (2) assuring that the actual performance equals predicted performance.

Process Improvement is (1) improving your capacity at each higher level of capability (more projects at the higher expectation levels) and (2) increasing your capability in each of the process areas (dotted lines).

the lower levels of a means is much greater than for higher levels. That translates into the risk that the prediction at low levels will be off. In analysis theory, these are prediction anomalies called outliers, which are ignored. In process improvement, these must be accommodated. There are some stellar software producers at CMM Level 1; therefore, at Level 1, the model is not fully predictive. However, the overwhelming odds are that Level 1 performance will be worse than the performance of organizations at higher levels.

The real key to process improvement is hidden in the first phrase of a sentence two paragraphs back: "If you know what to do, ..." Process improvement is not intuitive ("perfect practice makes perfect"). At risk of confusing my explanation of process improvement, the process that most needs implementing is the one you use for

process improvement. Once this process is good, the result will be an improved software engineering process. In other words, for process improvement to succeed, the skill level of the process improvement project team must be high, not the skill level in software engineering groups.

Most organizations do not have the requisite process improvement skills among their management or staff. No matter how motivated, facilitated, or well led, a team without the right skills is likely to fail to implement process improvement. If you are in a typical organization, process improvement has failed at least once. If you blamed the team, you were wrong—your expectations were unfounded. If you blamed process improvement merely because it is an art that is difficult to master, you were wrong, too—process improvement is alive and well and works great in the

right hands. If you brought together the right resources in the right place at the right time with the right management and right leadership, you probably did not read this far. Good Luck. ♦

About the Author



Steve Neuendorf is an independent management consultant. He has over 25 years measurement and process improvement experience, with over 15 years in software engineering process improvement. He has a bachelor's, a master's, and a doctorate from the University of Puget Sound.

Voice: 425-557-8747
E-mail: steve@serv.net
Internet: www.serv.net/~steve

Need Assistance with Software Process Improvement?

Call the SPI Hotline at 801-775-5555 ext. 3055 DSN 775-775-5555 ext. 3055 or E-mail us at spi@stsc1.hill.af.mil.

We can answer questions about various software process improvement (SPI) issues, including

- How to get started on SPI.
- Available SPI training.
- Assessments.
- CMM key process areas.
- SPI best practices.
- SPI return on investment.

Software Technology Support Center (STSC) SPI veterans are on call to answer questions and research your problems for up to one hour without charge. We can provide you with policy, process, and procedure templates from our STSC library. If you need in-depth assistance, we will refer you to the appropriate experts using our database of pre-qualified consultants from the STSC and external sources.

Several SEPG members responsible for leading OO-ALC/TIS to its Level 5 status now work in the STSC. Call the SPI Hotline for their consulting services.



Passing the CBA/IPI Torch

The Air Force Communications Agency (AFCA) at Scott Air Force Base, Ill. has transferred its responsibility for conducting Capability Maturity Model (CMM)-Based Appraisals for Internal Process Improvement (CBA/IPI) for Air Force organizations to the Software Technology Support Center (STSC) at Hill Air Force Base, Utah. The CBA/IPI is a method licensed by the Software Engineering Institute to assess an organization's capability to develop and maintain software. STSC consultants have experience in CMM-Based Appraisals that range from maturity Levels 2-5.

Organizations that need CBA/IPI assessments should call the STSC staff at 801-775-5555 ext. 3065 DSN 775-5555 ext. 3065 or E-mail spi@stsc1.hill.af.mil.

