# Three Dimensions of Process Improvement
## Part I: Process Maturity

Watts S. Humphrey
*Software Engineering Institute*

*This is the first part of a three-part article on methods of software process improvement that were developed at the Software Engineering Institute (SEI): the Capability Maturity Model (CMM)® for software, the Personal Software Process (PSP)℠, and the Team Software Process (TSP)℠. The CMM provides an overall framework to describe the activities software organizations need to do to consistently produce effective results; the PSP helps engineers use process principles in their personal work; the TSP shows integrated product teams how to use these processes to consistently produce quality products on aggressive schedules and for their planned costs. Each method provides important benefits; organizations will get the best results by using all three. Part I describes the CMM. Parts II and III (to appear in subsequent issues) will cover the PSP and TSP.*

The first issue to be addressed in any improvement program is "Why should we improve?" Regardless of how logical the improvement is and regardless of growing evidence of its benefits, organizations do not launch successful process improvement programs until they have a compelling business reason.

One of the most effective improvement motivators is customer pressure. In the case of the CMM, the U.S. Air Force asked the Software Engineering Institute (SEI) to devise an improved method to select software vendors. When we responded with the CMM, the defense industry then had a compelling business reason to improve their software processes. As the CMM became more widely used, the business logic for improvement also evolved. Now, many software organizations desire to establish and maintain a strong competitive position as quality suppliers of software-intensive systems.

Once the "why" question is answered, the next immediate question is "What must we do to achieve a superior software capability?" This is the principal question addressed by the CMM. The maturity-level framework and related evaluation system help organizations understand their capabilities. They can then compare their current practices with the CMM model and see what activities they need to add or improve.

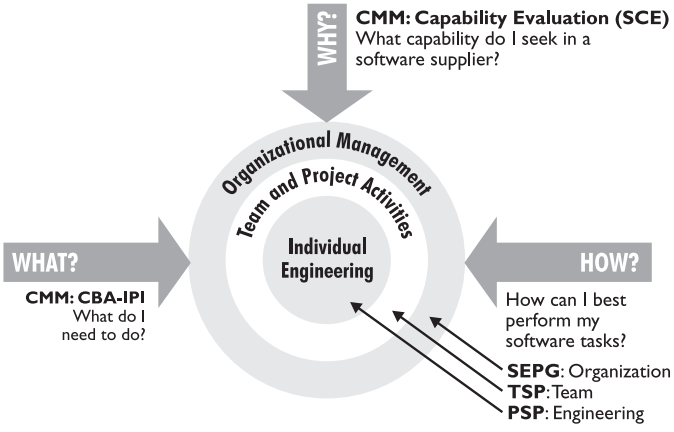
Once an organization knows what to do to improve, the next question is "How do we make these improvements?" This question involves several organizational levels. At the management level, a software engineering process group (SEPG) helps establish the definition, control, and improvement tasks needed to launch an improvement program. At the next level, the TSP guides project teams and their management in applying process principles to meet project objectives. Finally, the PSP addresses the way engineers develop products. For the organization to meet its objectives, the engineers and managers at all these levels must perform their tasks with skill and competence. In the last analysis, to do superior software work, organizations must have high-performance software engineers working on well-managed and high-performing project teams.

Figure 1 shows the relationships among these management, team, and engineering layers. Organizational capability is built from the inside out while project support and engineering motivation are provided from the outside in. Effective work in all three layers is required for a fully productive software organization. The balance of this article describes why we developed the CMM, PSP, and TSP, and how they work.

## Why Software Projects Fail

There are many ways to botch up a software project: have ineffective management, execute poor engineering, or create a confusing design. To be consistently successful, competent work must be done in all important technical and management areas. If only one area is not addressed—requirements management, for example—that area could easily become the Achilles heel that results in the project's failure. (Not to say that requirements management is more important than all the things done correctly, just that through neglect it became the cause of the failure). The next time around, requirements may

Figure 1. *Process improvement dimensions.*




*The SEI's work is supported by the Department of Defense.*

be managed properly but the design botched, and another project fails.

Although the cause of failure may not matter to the ultimate user, the reasons are important from a process improvement perspective. If the causes are managerial or administrative, improved organizational and project management will usually solve the problems. If the causes are purely technical, however, the solutions are generally more complex.

A careful examination of failed projects shows that they often fail for nontechnical reasons. The most common problems concern poor scheduling and planning or uncontrolled requirements. Poorly run projects also often lose control of changes or fail to use even rudimentary quality practices.

## The SEI Process Improvement Approach

The SEI started its process improvement work by first attacking project management problems, because poor management practices are the most frequent cause of software project failure. When the project is poorly managed, it is hard to improve anything else.

Again, it is important to remember that all aspects of the project are important, and any omitted area could be fatal. Thus, while an organization's improvement work focuses on management issues, the development groups must continue to do their best to address all the important facets of their work.

### Focusing the Improvement Effort

Process change is behavioral change and behavioral change takes a great deal of time. Behavior involves long-held habits and practices that people will not readily abandon. If change is imposed by force, people will resist what they perceive to be a threat to their stability. The most effective way to avoid resistance is to convince people that the change is in their best interest; however, it may take a long time for some people to be persuaded. Further, to have any hope of a lasting change, only a few changes should be addressed at a time. With several dozen critical issues, for example, making them all high priority would be confusing, and when confused, people are much less likely to change the way they work.

The only way to initiate process improvement is to look at the problems in *your* organization. No two organizations are identical, and you need to determine where your group can most productively focus its improvement energies. Figure out which of these problems are fundamental and which will make the most difference. Then, select those few changes that will most directly address the current needs of your business and people.

The CMM process maturity road map and the CMM-based appraisal for internal process improvement (CBA-IPI) helps organizations evaluate their particular weaknesses [1]. By doing a CBA-IPI assessment, organizations can both identify the most important areas for improvement and establish an organizational consensus on the need for improvement. This helps organizations set improvement priorities and reduces resistance to change [2, 3].

### The Motivation for the CMM

When I arrived at the SEI in 1986, I was asked to work on a high-priority Air Force project. The military's experience with software acquisition had been, in summary, less than satisfactory. This "software crisis" was the principal reason the U.S. Department of Defense (DoD) established the SEI in the first place. Thus, one of our first projects was to devise a method the DoD acquisition community could use to identify competent software contractors.

We knew from many years of software experience that management attention is the key to process improvement. Without sufficient management priority, not much improvement work gets accomplished, at least not for a long time. Management, however, is generally sensitive to their customers' demands. Thus, if the DoD acquisition community required improved software processes from their suppliers, we were sure management would give the subject a high priority. The original motivation for the CMM was thus to address DoD's problems with software acquisition.

### The Birth of the CMM Concept

The original concept behind the CMM started to gel in my mind many years ago when I was put in charge of a large software development organization. This group had several thousand engineers working on many large and small projects in 15 laboratories in the United States and Europe. Almost all the projects were in trouble and nothing was on schedule.

My first step was to visit several of the largest of these development laboratories. Nobody had plans or schedules. They all agreed that to deliver products on time, they needed to have schedules and plans. They also agreed that if they did not make plans, they could never make sensible commitments. And without sensible commitments, they could not start delivering on schedule. They said the reason they could not make plans and schedules was that if they put anyone on the planning work, they would have to take them from development, which would delay the projects.

So none of the laboratories made plans and schedules. Their commitments were pure guesses, and they were almost never met. This had been going on since the laboratories were formed, and it appeared likely to continue unless something happened to cause change. My question was if everybody agreed that planning was essential, why didn't they do it?

### It Was My Fault

The answer was that I was not doing *my* job. These laboratories had so much to do and they were under so much pressure that they could only do what they absolutely had to do to ship the products. In simplest terms, that meant coding and testing; everything else was considered unnecessary and could be skipped. As long as the laboratories could announce and ship products without planning, they would continue to do so.

Merely telling the laboratories to make plans would not work. So I put out a directive that henceforth, no prod-

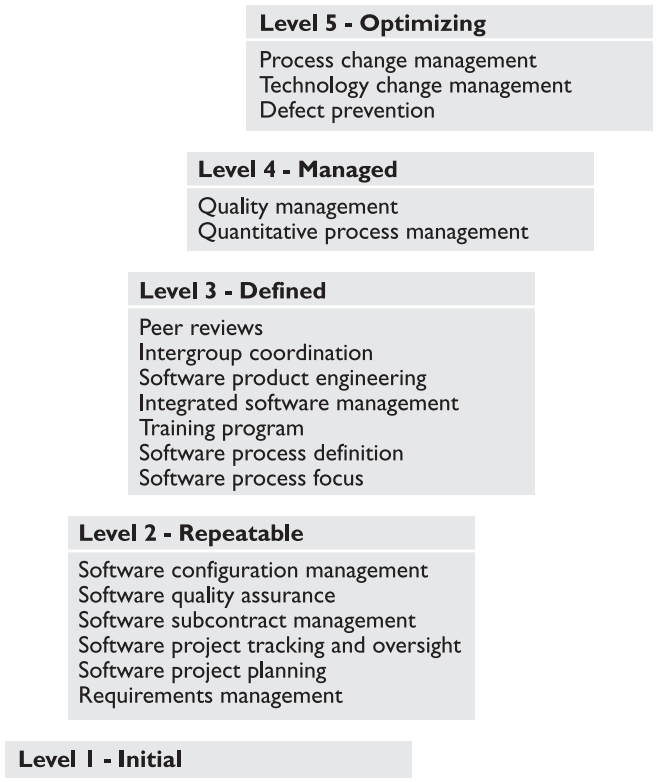



Figure 2. *CMM maturity levels.*

uct would be funded, committed, or shipped unless I first had a documented plan on my desk. This plan had to be signed by all the managers whose groups were involved in the work. I gave the laboratories 60 days to complete the first plans. Once we had plans for the work, we announced the new schedules, and we did not miss a single shipment date for the next two and one-half years.

### The First Important Lesson

Effective software process improvement will not start until management insists that product development work be planned and properly managed. Senior management must then continue to insist that all the projects be planned, even in a crisis. They not only must insist on aggressive plans, they must respect the plans and ensure that the engineers own and defend them. Since the schedules engineers initially make are almost always too tight, management must review and critique these plans to look for holes and oversights. While management should push as hard as they can for aggressive schedules, once the engineers have defended their plans, they should respect these plans and not override them with schedule edicts.

We had to start at the top with a management commitment to planning. The engineers could not take that step themselves. As long as senior management let them slide by without plans, they would continue to do so. Of course it was not quite that simple. The engineers needed a lot of help with their first plans, and we had to develop a project management training program. Over the next three years, we put 1,000 managers through a one-week project management course.

## Building the CMM Model

When I arrived at the SEI, we started to think about the improvement process in a more orderly way. We realized that process improvement must be taken in steps not only because people can change just a few things at a time but also because some steps are prerequisites for others. Since poor project management generally blocks good engineering, the first priority must be effective project management.

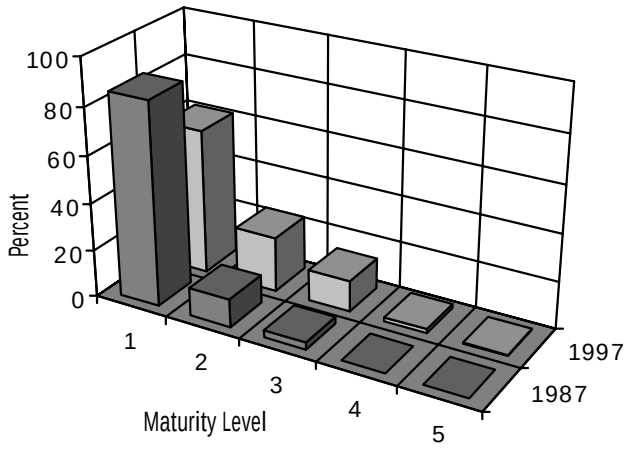
In addition to planning and scheduling, other tools that aid process improvement are quality assurance, configuration management, subcontract management, and requirements management. Quality assurance provides the management eyes and ears to ensure that the plans are properly made and that commitment ground rules are followed. Configuration management and subcontract management track and manage the products as they are developed, and effective requirements management maintains control of job scope. As shown in Figure 2, these are the CMM Level 2 practices [2, 3]. In essence, they establish a planned and managed environment that enables good engineering.

### The Higher Maturity Levels

Beyond basic management, the next step is to address organizational learning and growth. As long as organizations learn only from their own mistakes, they can expect to make a lot more mistakes. Most software organizations have many good practices that are only used by a few projects. An organization can make rapid improvement when these practices are more widely used. The objective is to capture those effective practices so others can use them, which leads to what we call process definition. Although there are many more parts to CMM Level 3, its essence is to facilitate organizational learning by getting good practices defined and into use.

The next steps focus on process measurements, quality management, and quality control. Organizations also need to introduce advanced methods and technologies. Finally, the

Figure 3. *State of software practice 1987-1997.*



| | |
|---|---|
| Yearly cost of improvement | $245,000 |
| Years engaged in improvement | 3.5 |
| Yearly cost per software engineer | $1,375 |
| Productivity gain per year | 35% |
| Yearly reduction in time to market | 19% |
| Yearly reduction in post-release defects | 39% |
| Business return per dollar invested | $5 |

Table 1. *Median results from an SEI study of 13 software organizations.*

software business is new, and we can expect it to continue to grow and evolve. Since any static process will soon be outdated, software organizations need to continue to improve. This calls for a continuing focus on technology change and process improvement. In total, these are the practices of CMM Levels 4 and 5.

### Introducing and Using the CMM

Once management is convinced of the business need for process improvement, the most effective way to start is to achieve a broad consensus on the need for and potential benefits of process improvement, which is the role of the CBA-IPI assessment. The SEI trains people on how to conduct such assessments, and many organizations now offer commercial assessment services. These offerings are described more fully on SEI's Web pages http://www.sei.cmu.edu.

## Conclusion

A large number of organizations have used the CMM framework, and a great deal has been published on its benefits [4,5,6,7,8,9,10,11]. Table 1 shows a brief summary of improvement data from one SEI study [12]. These 13 organizations had worked on software process improvement for an average of three and one-half years, and they all gained substantial cost, schedule, and quality benefits. Figure 3 shows the improvement in CMM levels between 1987—when SEI gathered initial data on a few dozen groups—and the most recent 1997 canvass of about 600 organizations. Although much more remains to be done, the CMM helps organizations improve. ◆

### About the Author

Watts S. Humphrey is a fellow at the SEI of Carnegie Mellon University, which he joined in 1986. At the SEI, he established the Process Program, led initial development of the CMM, introduced the concepts of Software Process Assessment and Software Capability Evaluation, and most recently, the PSP and TSP. Prior to joining the SEI, he spent 27 years with IBM in various technical executive positions, including management of all IBM commercial software development and director of programming quality and process.

He has master's degrees in physics from the Illinois Institute of Technology and in business administration from the University of Chicago. He is the 1993 recipient of the American Institute of Aeronautics and Astronautics Software Engineering Award. His most recent books include *Managing the Software Process* (1989), *A Discipline for Software Engineering* (1995), *Managing Technical People* (1996), and *Introduction to the Personal Software Process* (1997).

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Voice: 412-268-6379
E-mail: watts@sei.cmu.edu

### References

1. Dunaway, Donna and Steve Masters, "CMM-Based Appraisal for Internal Process Improvement (CBA-IPI): Method Description," Technical Report CMU/SEI 96TR-007, April 1996.
2. Humphrey, W. S., *Managing the Software Process*, Addison-Wesley, Reading, Mass., 1989.
3. Paulk, Mark C., et al., *The Capability Maturity Model: Guidelines for Improving the Software Process,* Addison-Wesley, Reading, Mass, 1995.
4. Diaz, Michael and Joseph Sligo, "How Software Process Improvement Helped Motorola," *IEEE Software*, Vol. 14, No. 5, Sept. 19, 1997, pp. 75-81.
5. Dion, Raymond, "Process Improvement and the Corporate Balance Sheet," *IEEE Software*, July 1993, pp. 28-35.
6. Goldenson, Dennis R. and James D. Herbsleb, "After the Appraisal: A Systematic Survey of Process Improvement, Its Benefits, and Factors that Influence Success," Technical Report CMU/SEI-95-TR-009, August 1995.
7. Hayes, Will and Dave Zubrow, "Moving On Up: Data and Experience Doing CMM-Based Process Improvement," Technical Report CMU/SEI-95-TR-008, August 1995.
8. Humphrey, W.S., T.R. Snyder, and R.R. Willis, "Software Process Improvement at Hughes Aircraft," *IEEE Software*, July 1991, pp. 11-23.
9. Lawlis, P.K., R.M. Flowe, and J.B. Thordahl, "A Correlational Study of the CMM and Software Development Performance." *CROSSTALK*, STSC, Hill Air Force Base, Utah, September 1995, pp. 21-25.
10. Wohlwend, H. and S. Rosenbaum, "Schlumberger's Software Improvement Program," *IEEE Transactions on Software Engineering*, Nov. 11, 1994, pp. 833-839.
11. Yamamura, George and Gary B. Wigle, "SEI CMM Level 5: For the Right Reasons," *CROSSTALK*, STSC, Hill Air Force Base, Utah, August 1997, pp. 3-6.
12. Herbsleb, J.D., D. Zubrow, D. Goldenson, W. Hayes, and M. Paulk, "Software Quality and the Capability Maturity Model," *Communications of the ACM*, June 1997, Vol. 40, No. 6, June 1997.