Effective CMM-Based Process Improvement

Mark C. Paulk, Software Engineering Institute, USA

Abstract
The Capability Maturity Model[SM] for Software developed by the Software Engineering Institute has had a major influence on software process and quality improvement around the world. Although the CMM[SM] has been widely adopted, there remain many misunderstandings about how to use it effectively for business-driven software process improvement. This paper discusses how to use the CMM correctly and effectively. It also discusses aspects of successful process improvement efforts that are not explicitly addressed by the CMM, but which are critical to achieving business and process improvement goals.

Keywords: CMM, Capability Maturity Model, software process, process improvement.

1. Introduction

The Capability Maturity Model[SM] for Software (CMM[SM] or SW-CMM[1]) developed by the Software Engineering Institute (SEI) has had a major influence on software process and quality improvement around the world [Paulk95]. The SW-CMM defines a five-level framework for how an organization matures its software process. These levels describe an evolutionary path from ad hoc, chaotic processes to mature, disciplined software processes. The five levels, and the 18 key process areas that describe them in detail, are summarized in Figure 1.

The five levels can be briefly described as:

1) Initial — The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.

This work is sponsored by the U.S. Department of Defense.

This paper was published as in **Proceedings of the 6th International Conference on Software Quality**, Ottawa, Canada, 28-31 October 1996, pp. 226-237.

[1] Because of the proliferation of capability maturity models inspired by the success of the CMM, we are starting to use the acronym SW-CMM when referring to the CMM for Software.

2) Repeatable	Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3) Defined	Management and engineering activities are documented, standardized, and integrated into a family of standard software processes for the organization. Projects use a tailored version of the organization's standard software processes for developing and maintaining software.
4) Managed	Detailed measures of the software process and product quality are collected. Software processes and products are quantitatively understood and controlled.
5) Optimizing	Continuous process improvement is facilitated by quantitative feedback from the process and from piloting innovative ideas and technologies.

The key process areas are satisfied by achieving goals, which are described by key practices, subpractices, and examples. The rating components of the SW-CMM are maturity levels, key process areas, and goals. The other components are informative and provide guidance on how to interpret the model.

Level	Focus	Key Process Areas
5 Optimizing	***Continuous process improvement***	**Defect Prevention** **Technology Change Management** **Process Change Management**
4 Managed	***Product and process quality***	**Quantitative Process Management** **Software Quality Management**
3 Defined	***Engineering processes and organizational support***	**Organization Process Focus** **Organization Process Definition** **Training Program** **Integrated Software Management** **Software Product Engineering** **Intergroup Coordination** **Peer Reviews**
2 Repeatable	***Project management processes***	**Requirements Management** **Software Project Planning** **Software Project Tracking & Oversight** **Software Subcontract Management** **Software Quality Assurance** **Software Configuration Management**
1 Initial	***Competent people and heroics***	

Figure 1. The key process areas in the SW-CMM.

While the SW-CMM has been very influential around the world in inspiring and guiding software process improvement (SPI), it has also been misused and

abused by some and not used effectively by others. Although the SEI's work has its critics [Bach94, Jones95], there is a growing body of data indicating the power of CMM-based process improvement [Herbsleb94, Lawlis95]. The purpose of this paper is to discuss the correct, effective use of the SW-CMM and to recommend specific software engineering and management practices that map to the SW-CMM, as noted in the footnotes. The recommendations in this paper reflect my personal opinion, based on my experience in working with a number of "world-class" software organizations.

Ideally, effective use of the SW-CMM should build an organization that can dynamically adapt to a rapidly changing, even chaotic, environment; an organization that knows what business it is in and pursues software projects aligned with its strategic business objectives; a learning organization that explicitly, rather than implicitly, captures knowledge; an organization managed by facts rather than intuition, while still valuing creativity; an organization that empowers its most crucial asset: its people.

2. Effective CMM-Based Process Improvement

Using the SW-CMM correctly means balancing conflicting objectives. CMM-based appraisals require the use of professional judgment. Although the SW-CMM provides a significant amount of guidance in making these judgments, removing subjectivity implies a deterministic, repetitive process that is not characteristic of engineering design work.

The SW-CMM is sometimes referred to as a set of process requirements, but it does not contain any "shall" statements. The goals, key practices, and subpractices of its key process areas are arranged in a hierarchy useful for interpreting or tailoring the SW-CMM. As Charlie Weber has pointed out, "To tailor a goal, you should sweat blood (and then be conservative in mapping the relationship to "the" CMM). To tailor a key practice, you should just sweat. To tailor a subpractice, your conscience should hardly bother you."

To a knowledgeable software professional, many of these recommendations may seem obvious. I fear, however, that the consistent implementation of even well-known good engineering and management practices is far from common.

2.1 Management Sponsorship

Trite though it may seem, obtaining senior management sponsorship is a crucial component of building organizational capability.[2] As individuals, we can exercise professionalism and discipline within our sphere of control, but if an organization as a whole is to change its performance, then its senior management

---

[2] Organization Process Focus, Commitment 2; Process Change Management, Commitment 2.

must actively support the change. Bottom-up SPI, without sponsorship and coordination, leads to islands of excellence rather than predictably improved organizational capability. Effective sponsorship will only occur, however, when there is significant dissatisfaction with the status quo.

Sponsorship involves such activities as setting policies[3] and providing resources for process work,[4] but the most crucial factor is that senior management believe in the strategic importance of process improvement, demonstrate their support, and pay attention.[5] If, on the other hand, management provides annual speeches on the importance of quality and then returns to the "really important" issues of cost and schedule, then that message will be received also.

"Paying attention" may lead to the Hawthorne Effect: people increase their efforts as a result of the attention, and productivity and quality could improve even if the work environment becomes worse. Sustaining improvement over an extended period, however, implies systematic change aligned with the business objectives of the organization.

If process improvement is a true priority, then management will monitor it closely.[6] Management should set aggressive improvement goals, while at the same time recognizing that achieving higher levels of software process maturity is incremental and requires a long-term commitment to continuous process improvement. It may be useful to view software process improvement as a project, with goals, plans for achieving those goals, and management of progress. Do not forget, however, that changing the culture requires the long view. CMM-based improvement is most effective within a systematic approach to SPI, such as the SEI's IDEAL[SM] approach.[7]

2.2 Commitments and Management by Fact

Management by fact is a paradigm shift for most organizations, which must be based on a measurement foundation.[8] To make data analysis useful, you need to understand what the data means and how to analyze it meaningfully. Begin by collecting a simple set of useful data (I recommend the Goal/Question/Metric approach [Basili92].). Then deploy standard definitions for typical measures[9] that support aggregating and comparing data from different environments.

---

3 Policy practices in Commitment to Perform.

4 Organization Process Focus, Abilities 1 and 2; in general, the resource practices in Ability to Perform.

5 Organization Process Focus, Commitment 3; management oversight practices in Verifying Implementation.

6 Organization Process Focus, Verification 1.

7 IDEAL stands for Initiate, Diagnose, Establish, Act, and Leverage.

8 Measurement practices in Measurement and Analysis.

9 Organization Process Definition, Activity 5.

A corollary to collecting data is to believe what it tells you. If you have historically developed 100 lines of code per month, "betting the company" that you can bring in a critical project at 1000 lines of code per month is almost certainly unwise. And changing the size estimates without changing commitments and functionality is not likely to help matters.

You also have to be sensitive to the potential for causing dysfunctional behavior by what you measure [Austin94]. People will focus their efforts where management is paying attention. If management only pays attention to schedule, then quality is likely to suffer. The act of measuring identifies what is important, but some things are difficult to measure. Management needs to ensure that attention is visibly paid to all critical aspects of the project, including those difficult to measure, not just those it is easy to measure and track. The corollary to believing the data is to recognize that no limited set of data can completely capture a complex operation.

Establish an internal commitment process[10] based on what you know you can do. If the workers buy in to a set of commitments as being realistic, even if aggressive, then they have made a personal commitment. If they consider the stated commitments as being a management pipe dream, they are likely to consider the resulting problems to be management's. This implies that management should seek worker feedback when establishing commitments.

2.3 Process Focus

In most organizations, a software engineering process group (SEPG) or some equivalent should be formed to coordinate process definition, improvement, and deployment activities.[11] The SEPG should be formed early, e.g., in the Initiating phase of the IDEAL approach, and be staffed with competent and respected individuals possessing both managerial and technical skills. It is crucial that these individuals have good interpersonal skills. Success of the SEPG depends on their ability to communicate, teach, negotiate, and consult [Mogilensky94].

One of the reasons for dedicating resources to an SEPG is to ensure follow-through on appraisal findings.[12] Many improvement programs have foundered simply because no action resulted from the appraisal. Improvement comes from action planning, assigning individuals to do the work, piloting and deploying improved processes, and management oversight throughout.

It is desirable to have part-time participants on process improvement teams. In saying that SEPGs should coordinate process activities, the word "coordinate"

---

10 Software Project Planning, Activities 1-3.

11 Organization Process Focus, Ability 1.

12 Organization Process Focus, Activity 1.

was deliberately chosen. The skills and knowledge of the organization's best people should be brought to bear on addressing its problems. In a world-class organization, I would expect everyone to participate in process and quality improvement activities.[13] Worker participation and empowerment enable successful process improvement.

Begin with the "as is" process, not the "should be" process, to leverage effective practices and co-opt resisters. Mandating top-down that everyone will follow the new "should be" process, particularly if not developed by empowered workers, is a common recipe for failure. The "as is" process evolved because the people doing the work needed to get the job done – even if that meant going around the system. The "should be" process may, or may not, be feasible in the given culture and environment. With an organizational focus on process management and improvement, the "as is" and "should be" processes will converge, resulting in organizational learning.

It is important to remember that software process improvement occurs in a business context. There may be many other crucial business issues being worked at the same time; there may even be a Total Quality Management (TQM) initiative under way. Since CMM-based improvement is an application of TQM principles to software, the synergy of aligning these initiatives seems obvious. Some of the most effective SPI programs I have seen operated under the umbrella of a TQM initiative – and in the instances where I have observed disjoint TQM and SPI programs, both were ineffective.

2.4 Useful Processes

Document your processes.[14] The reasons for documenting a process (or product) are 1) to communicate – to others now and perhaps to yourself later; 2) to understand – if you can't write it down, you don't really understand; and 3) to encourage consistency – take advantage of repeatability.

Building software is a design-intensive, creative activity. While the discipline of process is a crucial enabler of success, the objective is to solve a problem, and this requires creativity. Software processes should be repeatable, even if they are not repetitive. The balance between discipline and creativity can be challenging [Glass95]. Losing sight of the creative, design-intense nature of software work leads to stifling rigidity. Losing sight of the need for discipline leads to chaos.

Processes need to be tailored to the needs of the project [Ginsberg95].[15] Although standard processes provide a foundation, each project will also have

---

13 Process Change Management, Goal 2.

14 Organization Process Definition.

15 Organization Process Definition, Activity 4.

unique needs. Unreasonable constraints on tailoring can lead to significant resistance to following the process.

The following list contains some process attributes that are important for useful and effective processes:

- documented – but keep it simple
- trained – impart the skill to use it
- practiced – used even in a crisis
- measured – simple measures for insight into critical questions
- owned – by a responsible party
- maintained – updated as needed
- supported – in plans and by management expectations and rewards
- controlled – process changes are disciplined
- verified and validated – checked for correct execution

Keep the process simple because we live in a rapidly changing world. Processes do not need to be lengthy or complex. The SW-CMM is about doing things, not having things. A 1-2 page process description may suffice [Humphrey95], and subprocesses and procedures[16] can be invoked as needed and useful. Order is not created by complex controls, but by the presence of a few guiding formulae or principles [Wheatley92, page 11].

2.5 Training

Documented processes are of little value if they are not effectively deployed. Training, via a wide variety of mechanisms, is critical to consistent and effective software engineering and management.[17] Training is an investment in our greatest asset – our people [Curtis95].

The reason for training is to develop skills. There are many "training mechanisms" other than formal classroom training that can be effective in building skills. One that should be seriously considered is a formal mentoring program. In this case, formality means going beyond assigning a mentor and hoping that experience will rub off. Formality implies training people on how to mentor and monitoring the effectiveness of the mentoring. For example, the Onboard Shuttle project considers mentoring in doing causal analysis of defects [Paulk95].

---

16 Process descriptions are frequently composed of subprocesses, methods, and / or procedures, which in turn may be supported by automated tools. Procedures may be lengthy, but they are also usually deterministic.

17 Training Program.

Management training is particularly important because ineffective management can cripple a good team.[18] People who are promoted to management because of their technical skills have to acquire a new set of skills, including interpersonal skills [Mogilensky94, Weinberg94].

Having taken the Personal Software Process (PSP) course, I can highly recommend it for building self-discipline [Humphrey95]. Note that the effect of reading the book is not the same as taking the course and doing the work! Where the SW-CMM addresses the organizational side of process improvement, PSP addresses building the capability of individual practitioners. The PSP course convinces the individual, based on his or her own data, of the value of a disciplined, engineering approach to building software. Even if the organization is not interested in PSP, I would recommend it for professional development; applying PSP can be a survival skill in a level 1 organization.

2.6 Risk Management

Some argue that software project management is really risk management.[19] In one sense, the SW-CMM is about managing risk. We attempt to establish stable requirements[20] so that we can plan[21] and manage[22] effectively, but the business environment changes rapidly, perhaps chaotically. We try to establish an island of order in the sea of software chaos, but both order and chaos have a place. As Wheatley suggests, "To stay viable, open systems maintain a state of non-equilibrium, keeping the system in balance so that it can change and grow." [Wheatley92, page 78] Although we can establish processes that help us manage the risks of a chaotic world, we also need to change and grow.

This implies that you should use an incremental or evolutionary life cycle.[23] If you want to focus on risk management, the spiral model may be the preferred life cycle model. If you want to focus on involving the customer, perhaps rapid prototyping or joint application design would be preferable. Few long-term projects have the luxury of the stable environment necessary for the waterfall life cycle to be the preferred choice – yet it is probably the most common life cycle. (Note that for small projects, the waterfall life cycle may be an excellent choice.)

---

18 Software Project Planning, Ability 4; Software Project Tracking and Oversight, Ability 4; Integrated Software Management, Ability 3.

19 Software Project Planning, Activity 13; Software Project Tracking and Oversight, Activity 10; Integrated Software Management, Activity 10.

20 Requirements Management.

21 Software Project Planning.

22 Software Project Tracking and Oversight; Integrated Software Management.

23 Software Project Planning, Activity 5.

2.7 Customer-Supplier Relationship

Establish a good, working relationship with the customer and end user based on open communication and integrity.[24] This requires cooperation from the customer, and the danger of customer irrationality is difficult to overstate. Building good customer-supplier relationships is a long-term effort, but if the organization has (or wants to have) a stable customer base, the customer must appreciate the complexities of software engineering just as the supplier must understand the application domain and business environment for the software product. Even those organizations that operate in an environment where customer "churn" is the norm can profit from a (deserved) reputation for quality and integrity.

Talk to the customer. In the exercises for our Introduction to the CMM training, one of our most common observations is that the teams do not talk to their customer. This is crucial in making good cost, schedule, functionality, and quality tradeoff decisions. In a study done for one multinational company, which focused on leading-edge technology as its primary competitive advantage, the #1 and #2 issues for satisfying customer expectations were "quality" and "meeting commitments." "Technology" was #7.

One of the drivers behind software capability evaluations (SCE) is the customer's desire for predictable costs and schedules. Communication is critical for setting (and changing) customer expectations. Our contact at one of the early SCE pilot organizations commented later that he considered the improved customer-supplier communication to have been worth the (nontrivial) cost of doing SCEs, even if there had been no other improvement.

And, of course, be a good customer yourself when the tables are turned.

2.8 Peer Reviews

Although you can argue over the best kind of peer review, the simple fact is that the benefits of peer reviews far outweigh their costs.[25] The data suggests some form of inspection should be used [Ackerman89], but any form of collegial or disciplined review, such as structured walkthroughs, adds significant value.

Some time ago, one of my colleagues had an interesting question about peer reviews. He had been consulting with a company that had its supervisors review code. This had been successful for them and was firmly entrenched in their culture. His question was whether this satisfied the Peer Reviews key process

---

[24] Requirements Management; Software Project Tracking and Oversight, Activity 13; Software Quality Assurance, Activity 8; Intergroup Coordination, Goal 1, Activity 1.

[25] Peer Reviews.

area. My answer was that it did not. Peers are not supervisors, and having only one person do the review seems inadequate.

My colleague later reported that the company had performed an experiment comparing their supervisor reviews with peer reviews. They discovered that peer reviews found significantly more errors than supervisor reviews. They also discovered that supervisor reviews tended to identify "form errors" while peer reviews tended to identify "content errors." They also performed reviews with designers as well as "code-level peers" and found that these reviews identified more (and more design-oriented) errors than the code-level peer reviews.[26]

The significant point to me was that they experimented to find a data-based answer to an open issue. They did not take a consultant's advice on faith. I consider this one of the hallmarks of a learning organization, one that can take advantage of the SW-CMM effectively.

3. Conclusion

The SW-CMM represents a "common sense engineering" approach to software process improvement. Its maturity levels, key process areas, goals, and key practices have been extensively discussed and reviewed within the software community. While the SW-CMM is neither perfect nor comprehensive [Curtis95, Bate95], it does represent a broad consensus of the software community and is a useful tool for guiding SPI efforts.

Never forget why process improvement is important. Standards and models such as the SW-CMM can help organizations improve their software process, but focusing on achieving a maturity level without really improving the underlying process is a danger. Maturity levels should be a measure of improvement, not the goal of improvement. What are the business needs and business goals of the improvement effort? What is the impact on cost, cycle time, productivity, quality, and – most importantly – customer satisfaction?

4. References

Ackerman89 A.F. Ackerman, L.S. Buchwald, and F.H. Lewski, "Software Inspections: An Effective Verification Process," IEEE Software, Vol. 6, No. 3, May 1989, pp. 31-36.

Austin94 Robert D. Austin, "Theories of Measurement and Dysfunction in Organizations," PhD Dissertation, Department of Social and

[26] I think most peer review experts would agree that including designers in a peer review is a good idea. Including supervisors may be acceptable also, but there is a potential for dysfunction if they do performance reviews and are responsible for raises and promotions.

Decision Sciences, Carnegie Mellon University, 10 September 1994.

Bach94 James Bach, "The Immaturity of the CMM," American Programmer, Vol. 7, No. 9, September 1994, pp. 13-18.

Basili92 V.R. Basili, "Software Modeling and Measurement: The Goal / Question / Metric Paradigm," University of Maryland, CS-TR-2956, UMIACS-TR-92-96, 1992.

Bate95 Roger Bate, Dorothy Kuhn, Curt Wells, et al, "A Systems Engineering Capability Maturity Model, Version 1.1," Software Engineering Institute, CMU / SEI-95-MM-003, November 1995.

Curtis95 Bill Curtis, William E. Hefley, and Sally Miller, "People Capability Maturity Model," Software Engineering Institute, CMU / SEI-95-MM-02, September 1995.

Ginsberg95 Mark Ginsberg and Lauren Quinn, "Process Tailoring and the Software Capability Maturity Model," Software Engineering Institute, CMU / SEI-94-TR-024, November 1995.

Glass95 Robert L. Glass, **Software Creativity**, Prentice Hall, Englewood Cliffs, NJ, 1995.

Herbsleb94 James Herbsleb, Anita Carleton, et al., "Benefits of CMM-Based Software Process Improvement: Initial Results," Software Engineering Institute, CMU / SEI-94-TR-13, August 1994.

Humphrey95 Watts S. Humphrey, **A Discipline for Software Engineering**, ISBN 0-201-54610-8, Addison-Wesley Publishing Company, Reading, MA, 1995.

Jones95 Capers Jones, "The SEI's CMM – Flawed?," Software Development, Vol. 3, No. 3, March 1995, pp. 41-48.

Lawlis95 Patricia K. Lawlis, Robert M. Flowe, and James B. Thordahl, "A Correlational Study of the CMM and Software Development Performance," Crosstalk: The Journal of Defense Software Engineering, Vol. 8, No. 9, September 1995, pp. 21-25.

Mogilensky94 Judah Mogilensky and Betty L. Deimel, "Where Do People Fit in the CMM?," American Programmer, Vol. 7, No. 9, September 1994, pp. 36-43.

Paulk95 Carnegie Mellon University, Software Engineering Institute (Principal Contributors and Editors: Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis), **The Capability Maturity Model: Guidelines for Improving the Software Process**, ISBN 0-201-54664-7, Addison-Wesley Publishing Company, Reading, MA, 1995.

Weinberg94 Gerald M. Weinberg, **Quality Software Management, Volume 3: Congruent Action**, ISBN 0-932633-28-5, Dorset House, New York, NY, 1994.

Wheatley92 Margaret J. Wheatley, **Leadership and the New Science**, Berrett-Koehler Publishers, San Francisco, CA, 1992.

For Further Information

SEI Customer Relations
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-5800
Internet: customer-relations@sei.cmu.edu

The SEI Web page is
http://www.sei.cmu.edu/

For information specifically on the SW-CMM, visit
http://www.sei.cmu.edu/technology/cmm/

Acknowledgments

I would like to thank Donna Dunaway, Watts Humphrey, Susanne Kelly, Bill Peterson, Charlie Weber, and Dave Zubrow for their comments on this paper.