Using the Software CMM® in Small Organizations

Mark C. Paulk

Abstract

The Capability Maturity Model℠ 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 has been widely adopted, there remain many misunderstandings about how to use it effectively for business-driven software process improvement, particularly for small organizations and small projects. Some of the common problems with interpreting the Software CMM for the small project/organization include:

- What does "small" mean? In terms of people? Time? Size of project? Criticality of product?
- What are the CMM "requirements"? Are there key process areas or goals that should not be applied to small projects/organizations? Are there "invariants" of good processes?
- What are the drivers and motivations that cause abuse of the CMM?

This paper discusses how to use the CMM correctly and effectively in any business environment, with examples for the small organization. The conclusion is that the issues associated with interpreting the Software CMM for the small project or organization may be different in degree, but they are not different in kind, from those for any organization interested in improving its software processes. Using the Software CMM effectively and correctly requires professional judgment and an understanding of how the CMM is structured to be used for different purposes.

MARK C. PAULK
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213
Telephone: +1 (412) 268-5794
Fax: +1 (412) 268-5758
Internet: mcp@sei.cmu.edu

Mark is a Senior Member of the Technical Staff at the Software Engineering Institute. He has been with the SEI since 1987, initially working with the Software Capability Evaluation project. Mark has worked with the Capability Maturity Model project since its inception and was the project leader during the development of Version 1.1 of the Software CMM. He is also actively involved with software engineering standards, including

- ISO 15504 (aka SPICE -- Software Process Improvement and Capability dEtermination), an emerging suite of international standards for software process assessment
- ISO 12207, Software Life Cycle Processes
- ISO 15288, System Life Cycle Processes

Prior to joining the SEI, Mark was a Senior Systems Analyst for System Development Corporation (later Unisys Defense Systems) at the Ballistic Missile Defense Advanced Research Center in Huntsville, Alabama.

Mark received his master's degree in computer science from Vanderbilt University. He received his bachelor's degree in mathematics and computer science from the University of Alabama in Huntsville.

Professional society memberships and certifications

- Senior Member of the Institute of Electrical and Electronics Engineers (IEEE)
- Senior Member of the American Society for Quality (ASQ)
- ASQ Certified Software Quality Engineer

Using the Software CMM® in Small Organizations

Mark C. Paulk

1. Introduction

The Software Engineering Institute (SEI) is a federally funded research and development center established in 1984 by the U.S. Department of Defense with a broad charter to address the transition of software engineering technology – the actual adoption of improved software engineering practices. The SEI's existence is, in a sense, the result of the "software crisis" – software projects that are chronically late, over budget, with less functionality than desired, and of dubious quality. [Gibbs94] To be blunt, much of the crisis is self-inflicted, as when a Chief Information Officer says, "I'd rather have it wrong than have it late. We can always fix it later." The emphasis in many organizations on achieving cost and schedule goals, frequently at the cost of quality, once again teaches a lesson supposedly learned by American industry over twenty years ago and now enshrined in Total Quality Management (TQM).

To quote DeMarco [DeMarco95], this situation is the not-surprising result of a combination of factors:

- "People complain to us because they know we work harder when they complain."
- "The great majority [report] that their software estimates are dismal… but they weren't on the whole dissatisfied with the estimating process."
- "The right schedule is one that is utterly impossible, just not obviously impossible."

DeMarco goes on to observe that our industry is over-goaded, and the only real (perceived) option is to pay for speed by reducing quality.

The lesson of TQM is that focusing on quality leads to decreases in cycle time, increases in productivity, greater customer satisfaction, and business success. The challenge, of course, is defining what "focusing on quality" really means and then systematically addressing the quality issues. Perhaps the SEI's most successful product is the Capability Maturity Model for Software (CMM), a roadmap for software process improvement that has had a major influence on the software community around the world [Paulk95]. The Software 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 maturity levels prescribe priorities for successful process improvement, whose validity has been documented in many case studies and surveys [Herbsleb97, Lawlis95, Clark97].

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

Level	Focus	Key Process Areas
5 Optimizing	*Continual 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. An overview of the Software CMM.

Although the focus of the current release of the Software CMM, Version 1.1, is on large organizations and large projects contracting with the government, the CMM is written in a hierarchical form that runs from "universally true" abstractions for software engineering and project management to detailed guidance and examples. The key process areas in the CMM are satisfied by achieving goals, which are described by key practices, subpractices, and examples. The rating components of the CMM are maturity levels, key process areas, and goals. The other components are informative and provide guidance on how to interpret the model. There are 52 goals and 316 key practices for the 18 key process areas. Although the "requirements" for the CMM can be summarized in the 52 sentences that are the goals, the supporting material comprises nearly 500 pages of information. The practices and examples describe what good engineering and management practices are, but they are not prescriptive on how to implement the processes.

The CMM can be a useful tool to guide process improvement because it has historically been a common-sense application of Total Quality Management (TQM) concepts to software that was developed with broad review by the software community. Its five levels are simplistic, but when intelligently used they provide a lever for moving people such as the DOD program manager who bluntly stated, ""The bottom line is schedule. My promotions and raises are based on meeting schedule first and foremost."

While the Software CMM has been very influential around the world in inspiring and guiding software process improvement, it has also been misused and abused by some and not used effectively by others. The guidance provided by CMM v1.1 tends to be oriented towards large projects and large organizations. Small organizations find this problematic, although the fundamental concepts are, we believe, useful to any size organization in any application domain and for any business context.

Are meeting schedules, budgets, and requirements important to small projects? To small organizations? It is arguable that in some environments, such as the commercial shrinkwrap segment, cost is comparatively trivial when compared to the market share available to the first "good enough" product to ship. If the employees of an organization are satisfied with the status quo, there is little that the CMM can provide that will lead to true change; change occurs only when there is sufficient dissatisfaction with the status quo that managers and staff are willing to do things differently. This is as true for small organizations as large.

The CMM provides good advice on desirable management and engineering practices, with an emphasis on management, communication, and coordination of the human-centric, design-intensive processes that characterize software development and maintenance. It should be considered a guidebook

rather than a dictate, however, and the CMM user must apply professional judgment based on knowledge and experience in software engineering and management, plus the application domains and business environment of the organization. Because the CMM is focused on software, there are important aspects of TQM that are not directly addressed in the model, such as people issues and the broader perspective of systems engineering, which may also be crucial to the business. The CMM is a tool that should be used in the context of a systematic approach to software process improvement, such as the SEI's IDEAL model, illustrated in Figure 2 [McFeeley96].

An opening question for software process improvement discussions should always be: Why is the organization interested in using the Software CMM? If the desire is to improve process, with a direct tie to business objectives and a willingness to invest in improvement, then the CMM is a useful and powerful tool. If the CMM is simply the flavor of the month, then you have a prescription for disaster. If the driver is customer concerns, ideally the concerns will lead to collaborative improvement between customer and supplier. Sometimes the supplier's concern centers on software capability evaluations (SCEs), such as are performed by government acquisition agencies in source selection and contract monitoring. DOD policies on the criteria for performing SCEs would exclude most small organizations and small projects [Barbour96], but there are circumstances under which they may occur.

Many of the abuses of the Software CMM spring out of a fear of what "others" may do. If an organization applies common sense to the guidance in the CMM as guidance rather than requirements, then many of the interpretation problems of the model vanish. There are cases, however, where ignorance of good engineering and management practices is the problem. This is particularly problematic for good technical people who have been promoted into management positions, but who have little management experience or training. This contributes to the problems identified by a DOD task force [DOD87]:

- "Few fields have so large a gap between best current practice and average current practice."
- "The big problem is not technical... today's major problems with military software development are not technical problems, but management problems."

2. Small Organizations and Small Projects

The focus of this paper is on using the Software CMM correctly and effectively for small organizations because I am frequently asked, "Can the Software CMM be used for small projects (or small organizations)?" Yet the definition of "small" is challengingly ambiguous, as illustrated in Table 1. At one time there was an effort to develop a tailored CMM for small projects and organizations, but the conclusion of a 1995 CMM tailoring workshop was that we could not even agree on what "small" really meant! The result was a report on how to tailor the CMM rather than a tailored CMM for small organizations [Ginsberg95]. In a 1998 SEPG conference panel on the CMM and small projects [Hadden98a], small was defined as "3-4 months in duration with 5 or fewer staff." Brodman and Johnson define a small organization as fewer than 50 software developers and a small project as fewer than 20 developers [Johnson98].

Table 1. Defining a "Small" Project

Variant of "Small"	Number of People	Amount of Time
Small	3-5	6 months
Very small	2-3	4 months
Tiny	1-2	2 months
Individual	1	1 week
Ridiculous!	1	1 hour

Note that small to tiny projects are in the range being addressed by Humphrey in his Team Software Process^{SM} (TSP) work, and the individual effort is in the range of the Personal Software Process^{SM} (PSP) [Humphrey95]. TSP and PSP illustrate how CMM concepts are being applied to small projects. The "ridiculous" variant represents an interpretational problem. On the two occasions this variant has been discussed, the problem was the definition of "project." In both cases it was a maintenance environment,

and the organization's "projects" would have been described as tasks in the CMM; the more accurate interpretation for a CMM "project" was a baseline upgrade or maintenance release… but the terminology clash was confusing.

One of the first challenges for small organizations in using the CMM is that their primary business objective is to survive! Even after deciding the status quo is unsatisfactory and process improvement will help, finding the resources and assigning responsibility for process improvement, and then following through by defining and deploying processes is a difficult business decision. The small organization tends to believe

- we are all competent – people were hired to do the job, and we can't afford training in terms of either time or money
- we all communicate with one another – "osmosis" works because we're so "close"
- we are all heroes – we do whatever needs to be done, the rules don't apply to us (they just get in the way of getting the job done), we live with short cycle times and high stress

Yet small organizations, just like large ones, will have problems with undocumented requirements, the mistakes of inexperienced managers, resource allocation, training, peer reviews, and documenting the product. Despite these challenges, small organizations can be extraordinarily innovative and productive. Although there are massive problems that may require large numbers of people to solve, in general small teams are more productive than large teams – they jell quicker and there are far fewer communication problems. The question remains, however, is process discipline needed for small teams? To answer this CMM mantra, we need to consider what discipline involves – and that leads to the heart of this paper's CMM interpretation discussion.

One last precursor, however. When assessing "small" organizations, it is advisable to use a streamlined assessment process; the formality of a two-week CMM-based appraisal for internal process improvement (CBA IPI) is probably excessive [Strigel95, Paquin98, Williams98]. The emphasis should be on efficiently identifying important problems, even if some are missed due to lack of rigor. I recommend focusing on the institutionalization practices that establish the organization's culture: planning, training, etc.; and explicitly tying process improvement to business needs.

3. Interpreting the CMM

Where does the Software CMM apply? The CMM was written to provide good software engineering and management practices for any project in any environment. The model is described in a hierarchy

Maturity levels (5)
→ **Key process areas** (18)
→ **Goals** (52)
→ *Key practices* (316)
® *Subpractices and examples* (many)

In my experience over the last decade of software process work, environments where interpretation and tailoring of the CMM are needed include:

- very large programs
- virtual projects or organizations
- geographically distributed projects
- rapid prototyping projects
- research and development organizations
- software services organizations
- small projects and organizations

The interpretation guidance for small projects and small organizations is also applicable to large projects and organizations. Intelligence and common sense are required to use the CMM correctly and effectively [Paulk96]. It is simultaneously true that all (software) projects are different and all (software) projects are the same. We are required to balance conflicting realities: similarity versus uniqueness, order

versus chaos. Those who succeed build lasting organizations [Collins94] that are truly capable of organizational learning [Senge90]; the rest must derive their success elsewhere.

The "normative" components of the CMM are maturity levels, key process areas, and goals. All practices in the CMM are informative. Since the detailed practices primarily support large, contracting software organizations, they are not necessarily appropriate, as written, for direct use by small projects and small organizations – but they do provide insight into how to achieve the goals and implement repeatable, defined, measured, and continually improving software processes. Thus we prevent such "processes" as the estimating procedure that was simply "Go ask George."

My most frequent interpretation recommendation is to develop a mapping between CMM terminology and the language used by the organization. In particular, terms dealing with organizational structures, roles and relationships, and formality of processes need to be mapped into their organizational equivalents to prevent misunderstandings such as the "ridiculous one-hour project." Examples of organizational structures include "independent groups" such as quality assurance, testing, and configuration management. Appropriate organizational terminology for roles such as project manager and project software manager should be specified. People may fill multiple roles; for example, one person may be the project manager, project software manager, SCM manager, etc. Explicitly stating this makes interpretation of the CMM much simpler and more consistent.

Once the terminology issues are understood, we can think about what the "invariants" for a disciplined process are and which practices depend on the context. In general we assume that key process areas and goals are always relevant to any environment, with the exception of *Software Subcontract Management*, which may be "not applicable" if there is no subcontracting. In contrast, I can conceive of no circumstances under which *Peer Reviews* can be reasonably tailored out for a Level 3 organization. This is a matter of competent professional judgment, although an alternative practice such as formal methods might replace peer reviews. Professional judgment and trained, experienced assessors are crucial, even for small organizations! [Abbott97]

I have never seen an environment where the following were not needed (though implementations differ):

- documented customer (system) requirements
- communication with customer (and end users)
- agreed-to commitments
- planning
- documented processes
- work breakdown structure

Some practices, however, deal with "large-project implementations." A small project is unlikely to need an SCM group or a Change Control Board… but configuration management and change control are always necessary. An independent SQA group may not be desirable, but objective verification that requirements are satisfied always is. An independent testing group may not be established, but testing is always necessary. We thus see that even for context-sensitive practices, the intent is critical even if the implementation is radically different between small organizations and large. Many of the context-sensitive, large-project implementation issues relate to organizational structure. If one reads the CMM definition of "group," it states that "a group could vary from a single individual assigned part time, to several part-time individuals assigned from different departments, to several individuals dedicated full time," which is intended to cater to a variety of contexts.

In addition to these, specific questions that arise repeatedly, especially for small organizations, relate to:

- management sponsorship
- measurement
- SEPGs
- "as is" processes
- documented processes

- tailoring
- training
- risk management
- planning
- peer reviews

Trite though it may seem, obtaining senior management sponsorship is a crucial component of building organizational capability. 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 must actively support the change. Bottom-up improvement, without sponsorship and coordination, leads to islands of excellence rather than predictably improved organizational capability. It should be noted, however, that for small organizations, while the president (or founder) is the primary role model, a respected "champion" frequently has the influence to move the entire organization – including the president.

Management by fact is a paradigm shift for most organizations, which must be based on a measurement foundation. 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. You also have to be sensitive to the potential for causing dysfunctional behavior by what you measure [Austin96]. 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.

In most organizations, a software engineering process group (SEPG) or some equivalent should be formed to coordinate process definition, improvement, and deployment activities. One of the reasons for dedicating resources to an SEPG is to ensure follow-through on appraisal findings. Many improvement programs have foundered simply because no action resulted from the appraisal. Small organizations may not have full-time SEPG staff, but the responsibility for improvement should be explicitly assigned and monitored.

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.

Document your processes. 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. Documented processes support organizational learning and prevent reinventing the wheel for common problems – they put repeatable processes in place. Documentation is therefore important, but documents need not be lengthy or complex to be useful. Keep the process simple because we live in a rapidly changing world. Processes do not need to be lengthy or complex. The CMM is about doing things, not having things. A 1-2 page process description may suffice, and subprocesses and procedures can be invoked as needed and useful. Use good software design principles, such as locality, information hiding, and abstraction, in defining processes. Another useful rule of thumb is to track work at 2-3 tasks per week at most. Order is not created by complex controls, but by the presence of a few guiding formulae or principles [Wheatley92, page 11].

Processes need to be tailored to the needs of the project [Ginsberg95, Ade96]. Although standard processes provide a foundation, each project will also have unique needs. Unreasonable constraints on tailoring can lead to significant resistance to following the process. As Hoffman expresses it, "Don't require processes that don't make sense." [Hoffman98]

The degree of formality needed for processes is a frequent challenge for both large and small organizations [Comer98]. Should there be separate procedure for each of the 25 key practices at Level 2 that mention "according to a documented procedure?" [Hadden98a, Pitterman98] The answer, as discussed in section 4.5.5 "Documentation and the CMM" of the CMM book [Paulk95], is a resounding NO! Packaging of documentation is an organizational decision.

Documented processes are of little value if they are not effectively deployed. To achieve buy-in for the documented, process implementers must be part of process definition and improvement. Training, via a wide variety of mechanisms, is critical to consistent and effective software engineering and management. 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.

Training remains an issue after the initial deployment of a process or technology [Abbott97, Williams98]. As personnel change, the incremental need for training may not be adequately addressed. Mentoring and apprentice programs may suffice to address this issue, but they cannot be assumed to be satisfactory without careful monitoring.

Management training is particularly important because ineffective management can cripple a good team. People who are promoted to management because of their technical skills have to acquire a new set of skills, including interpersonal skills [Mogilensky94, Curtis95, Weinberg94].

Some argue that software project management is really risk management. In one sense, the CMM is about managing risk. We attempt to establish stable requirements so that we can plan and manage 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. 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, however, that for small projects, the waterfall life cycle may be an excellent choice.

The #1 factor in successful process definition and improvement is "planfulness" [Curtis96]. Planning is needed for every major software process, but within the bounds of reasonable judgment, the organization determines what is "major" and how the plan should be packaged. A plan may reside in several different artifacts or be embedded in a larger plan.

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. 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. Recognizing the value of peer reviews does not mean, unfortunately, that we do them systematically. We need to "walk the walk," not just "talk the talk." This is very frustrating for technical people who do not understand the emphasis on management in the CMM, yet poor management leads to abandoning good engineering practices such as peer reviews.

There are other issues that have been identified for small organizations and projects. Paquin [Paquin98] identifies five:

- assessments
- project focus

- documentation
- required functions
- maturity questionnaire

We have not discussed the project focus of Level 2 as being a challenge for small organizations. Software process improvement involves overhead that may be excessive for a small project. Some recommend attacking small project process improvement from an organizational perspective [Comer98, Paquin98], which is certainly a reasonable approach, even it does seem to mix Levels 2 and 3. This is a consideration for any size organization or project [Paulk96]. Although an organization can achieve Level 2 without an organization process focus, the most effective organizational learning strategy will be one that stresses organizational assets that lessen the overhead of projects. At the same time, it must be recognized that there may be resistance to change at the project level, perhaps based on valid concerns, and addressing resistance needs to be considered part of the organization's learning process.

Required functions are an issue because there may be more CMM functions than there are people. This issue has been discussed as terminology or role mapping. The maturity questionnaire is a concern because it uses CMM terminology, thus it may be confusing to those filling it out. Expressing the questionnaire in the terminology of the organization is thus a desirable precursor to even an informal assessment or survey.

Abbott [Abbott97] identifies six keys to software process improvement in small organizations:
- senior management support
- adequate staffing
- applying project management principles to process improvement
- integration with ISO 9001
- assistance from process improvement consultants
- focus on providing value to projects and to the business

If applying good project management to software projects is the best way to ensure success, then the same should be true for process improvement, which should be treated like any other project. ISO 9001 is more frequently an issue for large organizations than small, so it is interesting that Abbott points this out for his small company.

Brodman and Johnson [Johnson98] identify seven small organization/small project challenges:
- handling requirements
- generating documentation
- managing projects
- allocating resources
- measuring progress
- conducing reviews
- providing training

Brodman and Johnson have developed a tailored version of the CMM for small businesses, organizations, and projects [Johnson96, Johnson97, Brodman94]. Although the majority of the key practices in the CMM were tailored in the LOGOS Tailored CMM, the changes can be characterized as:
- clarification of existing practices
- exaggeration of the obvious
- introduction of alternative practices (particularly as examples)
- alignment of practices with small business/small organization/small project structure and resources

Therefore the changes involved in tailoring the CMM for small organizations should not be considered radical.

4. Abusing the Software CMM

Using the CMM correctly means balancing conflicting objectives. CMM-based appraisals require the use of professional judgment. Although the 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 CMM is sometimes referred to as a set of process requirements, but it does not contain any "shall" statements. That is why it is an abuse of the CMM to check off (sub)practices for conformance.

Some are unwilling or unable to interpret, tailor, or apply judgment. It is easy to mandate the key practices, but foolhardy. This foolishness is frequently driven by paranoia about customer intentions and competence. On more than one occasion I have heard someone say they were doing something that was foolish, but they ware afraid that the customer was so ignorant or incompetent that they would be unable to understand the rationale for doing things differently than literally described in the CMM. This is particularly problematic for SCEs. It is true that judgments may differ – and sometimes legitimately so. What is adequate in one environment may not suffice for a new project. That is why we recommend that process maturity be included in risk assessment rather than using maturity levels to filter offerors [Barbour96]. Small organizations should have less of a concern with this problem since it is unlikely that SCEs for small organizations are cost-effective. It is more of a problem for large organizations with many small projects.

Unfortunately I have no solution for this problem. "Standards" such as the CMM can help organizations improve their software process, but focusing on achieving a maturity level without addressing the underlying process can cause dysfunctional behavior. Maturity levels should be measures of improvement, not goals of improvement. That is why we emphasize the need to tie improvement to business objectives.

5. Conclusion

The bottom line is that software process improvement should be done to help the business – not for its own sake. This is true for both large organizations and small. The best advice comes from Sanjiv Ahuja, President of Bellcore: "Let common sense prevail!"

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.

The 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 CMM is neither perfect nor comprehensive, it does represent a broad consensus of the software community and is a useful tool for guiding improvement efforts, and it can be uses to help small software organizations improve their processes [Abbott97, Hadden98b, Hoffman98, Pitterman98, Sanders98].

Small organizations should seriously consider PSP and TSP [Ferguson97, Hayes97]. Having taken the PSP course, I can highly recommend it for building self-discipline. Note that the effect of reading the book is not the same as taking the course and doing the work! Where the 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.

References

Abbott97 John J. Abbott, "Software Process Improvement in a Small Commercial Software Company," , **Proceedings of the 1997 Software Engineering Process Group Conference**, San Jose, CA, 17-20 March 1997.

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.

Ade96 Randy W. Ade and Joyce P. Bailey, "CMM Lite: SEPG Tailoring Guidance for Applying the Capability Maturity Model for Software to Small Projects," **Proceedings of the 1996 Software Engineering Process Group Conference: Wednesday Papers**, Atlantic City, NJ, 20-23 May 1996.

Austin96 Robert D. Austin, **Measuring and Managing Performance in Organizations**, Dorset House Publishing, ISBN: 0-932633-36-6, New York, NY, 1996.

Barbour96 Rick Barbour, “Software Capability Evaluation Version 3.0 Implementation Guide for Supplier Selection,” Software Engineering Institute, Carnegie Mellon University, CMU/SEI-95-TR-012, April 1996.

Brodman94 J.G. Brodman and D.L. Johnson, "What Small Businesses and Small Organizations Say About the CMM," **Proceedings of the 16th International Conference on Software Engineering**, IEEE Computer Society Press, Sorrento, Italy, 16-21 May 1994, pp. 331-340.

Clark97 Bradford K. Clark, "The Effects of Software Process Maturity on Software Development Effort," PhD Dissertation, Computer Science Department, University of Southern California, August 1997.

Collins94 James C. Collins and Jerry I. Porras, **Built to Last**, HarperCollins Publishers, New York, NY, 1994.

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

Curtis96 Bill Curtis, "The Factor Structure of the CMM and Other Latent Issues," **Proceedings of the 1996 Software Engineering Process Group Conference: Tuesday Presentations**, Atlantic City, NJ, 20-23 May 1996.

DeMarco95 Tom DeMarco, **Why Does Software Cost So Much?**, ISBN 0-932633-34-X, Dorset House, New York, NY, 1995.

DOD87 Department of Defense, "Report of the Defense Science Board Task Force on Military Software," Office of the Under Secretary of Defense for Acquisition, Washington, D.C., September 1987.

Ferguson97 Pat Ferguson and Jeanie Kitson, “CMM-Based Process Improvement Supplemented by the Personal Software Process in a Small Company Environment,” , **Proceedings of the 1997 Software Engineering Process Group Conference**, San Jose, CA, 17-20 March 1997.

Gibbs94 W. Wayt Gibbs, "Software's Chronic Chrisis," Scientific American, September 1994, pp. 86-95.

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.

Hadden98a Rita Hadden, “How Scalable are CMM Key Practices?” Crosstalk: The Journal of Defense Software Engineering, Vol. 11, No. 4, April 1998, pp. 18-20, 23.

Hadden98b Rita Hadden, "Key Practices to the CMM: Inappropriate for Small Projects?" panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.

Hayes97 Will Hayes and James W. Over, "The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-97-TR-001, December 1997.

Herbsleb97 James Herbsleb, David Zubrow, Dennis Goldenson, Will Hayes, and Mark Paulk, "Software Quality and the Capability Maturity Model," Communications of the ACM, Vol. 40, No. 6, June 1997, pp. 30-40.

Hoffman98 Leo Hoffman, "Small Projects and the CMM," in "Key Practices to the CMM: Inappropriate for Small Projects?" panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.

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

Johnson96 Donna L. Johnson and Judith G. Brodman, **The LOGOS Tailored Version of the CMM for Small Businesses, Small Organizations, and Small Projects**, Version 1.0, August 1996.

Johnson97 Donna L. Johnson and Judith G. Brodman, "Tailoring the CMM for Small Businesses, Small Organizations, and Small Projects," Software Process Newsletter, IEEE Computer Society Technical Council on Software Engineering, No. 8, Winter 1997, p. 1-6.

Johnson98 Donna L. Johnson and Judith G. Brodman, "Applying the CMM to Small Organizations and Small Projects," **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.

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. Reprinted in Software Process Newsletter, IEEE Computer Society Technical Council on Software Engineering, No. 7, Fall 1996, pp. 1-5.

McFeeley96 Bob McFeeley, "IDEAL: A User's Guide for Software Process Improvement," Software Engineering Institute, CMU/SEI-96-HB-001, February 1996.

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.

Paquin98 Sherry Paquin, "Struggling with the CMM: Real Life and Small Projects," in "Key Practices to the CMM: Inappropriate for Small Projects?" panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.

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.

Paulk96 Mark C. Paulk, "Effective CMM-Based Process Improvement," **Proceedings of the 6th International Conference on Software Quality**, Ottawa, Canada, 28-31 October 1996, pp. 226-237.

Pitterman98 Bill Pitterman, "Key Practices to the CMM: Inappropriate for Small Projects?" panel, Rita Hadden moderator, **Proceedings of the 1998 Software Engineering Process Group Conference**, Chicago, IL, 9-12 March 1998.

Sanders98 Marty Sanders, "Small Company Action Training and Enabling," in **The CMM and Small Projects**, Society for Software Quality Roundtable, Washington, DC, 26 January 1998.

Senge90 Peter M. Senge, **The Fifth Discipline: The Art & Practice of the Learning Organization**, Doubleday/Currency, New York, NY, 1990.

Strigel95 Wolfgang B. Strigel, "Assessment in Small Software Companies," **Proceedings of the 1995 Pacific Northwest Software Quality Conference**, 1995, pp. 45-56.

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.

Williams98 Louise B. Williams, "SPI Best Practices for 'Small' Projects," in **The CMM and Small Projects**, Society for Software Quality Roundtable, Washington, DC, 26 January 1998.