The Immaturity of CMM
by James Bach
(Formerly of Borland International)
This article was originally published in the September =9194 issue of American Programmer.

The Software Engineering Institute's (SEI) Capability Maturity Model (CMM) gets a lot of publicity. Given that the institute is funded by the US Department of Defense to the tune of tens of millions of dollars each year [1], this should come as no surprise=97 the folks at the SEI are the official process mavens of the military, and have the resources to spread the word about what they do. But, given also that the CMM is a broad, and increasingly deep, set of assertions as to what constitutes good software development practice, it's reasonable to ask where those assertions come from, and whether they are in fact complete and correct.

My thesis, in this essay, is that the CMM is a particular mythology of software process evolution that cannot legitimately claim to be a natural or essential representation of software processes.

The CMM is at best a consensus among a particular group of software engineering theorists and practitioners concerning a collection of effective practices grouped according to a simple model of organizational evolution. As such, it is potentially valuable for those companies that completely lack software savvy, or for those who have a lot of it and thus can avoid its pitfalls.

At worst, the CMM is a whitewash that obscures the true dynamics of software engineering, suppresses alternative models. If an organization follows it for its own sake, rather than simply as a requirement mandated by a particular government contract, it may very well lead to the collapse of that company's competitive potential. For these reasons, the CMM is unpopular among many of the highly competitive and innovative companies producing commercial shrink-wrap software.

A short description of the CMM
The CMM [7] was conceived by Watts Humphrey, who based it on the earlier work of Phil Crosby. Active development of the model by the SEI began in 1986.
It consists of a group of "key practices", neither new nor unique to CMM, which are divided into five levels representing the stages that organizations should go through on the way to becoming "mature". The SEI has defined a rigorous process assessment method to appraise how well a organization satisfies the goals associated with each level. The assessment is supposed to be led by an authorized lead assessor.

The maturity levels are:

1. Initial (chaotic, ad hoc, heroic)
2. Repeatable (project management, process discipline)
3. Defined (institutionalized)

4. Managed (quantified)
5. Optimizing (process improvement)

One way companies are supposed to use the model is first to assess their maturity level and then form a specific plan to get to the next level. Skipping levels is not allowed.

The CMM was originally meant as a tool to evaluate the ability of government contractors to perform a contracted software project. It may be suited for that purpose; I don't know. My concern is that it is also touted as a general model for software process improvement. In *that* application, the CMM has serious weaknesses.
Shrink-wrap companies, which have also been called commercial off-the-shelf firms or software package firms, include Borland, Claris, Apple, Symantec, Microsoft, and Lotus, among others. Many such companies rarely if ever manage their requirements documents as formally as the CMM describes. This is a requirement to
achieve level 2, and so all of these companies would probably fall into level 1 of the model.

Criticism of the CMM

A comprehensive survey of criticism of the CMM is outside the scope of this article. However, Capers Jones and Gerald Weinberg are two noteworthy critics.
In his book *Assessment & Control of Software Risks* [6], Jones discusses his own model, Software Productivity Research (SPR), which was developed independently from CMM at around the same time and competes with it today. Jones devotes a chapter to outlining the weaknesses of the CMM. SPR accounts for many factors that the CMM currently ignores, such as those contributing to the productivity of individual engineers.
In the two volumes of his *Quality Software Management* series [12,13], Weinberg takes issue with the very concept of maturity as applied to software processes, and instead suggests a paradigm based on *patterns* of behavior. Weinberg models software processes as interactions between humans, rather than between formal constructs. His approach suggests an evolution of "problem-solving leadership" rather than canned processes.

General problems with CMM
I don't have the space to expand fully on all the problems I see in the CMM. Here are the biggest ones from my point of view as a process specialist in the shrink-wrap world:

=B7 The CMM has no formal theoretical basis. It's based on the experience of "very knowledgeable people". Hence, the de facto underlying theory seems to be that experts know what they're doing. According to such a principle, any other model based on experiences of other knowledgeable people has equal veracity.

=B7 The CMM has only vague empirical support. That is, the empirical support for CMM could also be construed to support other models. The model is based mainly on experience of large government contractors, and Watts Humphrey's own experience in the mainframe world. It does not account for the success of shrink-wrap companies, and levels 1, 4, and 5 are not well represented in the data: the first because it is misrepresented, the latter two because there are so few organizations at those levels. The

SEI's, Mark Paulk can cite numerous experience reports supporting CMM, and he tells me that a formal validation study is underway. That's all well and good, but the anecdotal reports I've seen and heard regarding success using the CMM could be interpreted as evidence for the success of people working together to achieve *anything*. In other words, without a *comparison* of alternative process models under controlled conditions, the empirical case can never be closed. On the contrary, the case is kept wide open by ongoing counterexamples in the form of successful level 1 organizations, and by the curious lack of data regarding failures of the CMM (which may be due to natural reluctance on the part of companies to dwell on their mistakes, or of the SEI to record them).

=B7 The CMM reveres process, but ignores people. This is readily apparent to anyone who is familiar with the work of Gerald Weinberg, for whom the problems of human interaction define engineering. By contrast, both Humphrey and CMM mention people in passing [5], but both also decry them as unreliable and assume that defined processes can somehow render individual excellence less important. The idea that process makes up for mediocrity is a pillar of the CMM, wherein humans are apparently subordinated to defined processes. But, where is the justification for this? To render excellence less important the problem solving tasks would somehow have to be embodied in the process itself. I've never seen such a process, but if one exists, it would have to be quite complex. Imagine a process definition for playing a repeatably good chess game. Such a process exists, but is useful only to computers; a process useful to humans has neither been documented nor taught as a series of unambiguous steps. Aren't software problems at least as complex as chess problems?

=B7 The CMM reveres institutionalization of process for its own sake. Since the CMM is principally concerned with an organization's ability to commit, such a bias is understandable. But, an organization's ability to commit is merely an expression of a project team's ability to execute. Even if necessary processes are not institutionalized formally, they may very well be in place, informally, by virtue of the skill of the team members. Institutionalization guarantees nothing, and efforts to institutionalize often lead to a bifurcation between an oversimplified public process and a rich private process that must be practiced undercover. Even if institutionalization is useful, why not instead institutionalize a system for identifying and keeping key contributors in the organization, and leave processes up to them?

=B7 The CMM contains very little information on process dynamics. This makes it confusing to discuss the relationship between practices and levels with a CMM proponent, because of all the hidden assumptions. For instance, why isn't training on level 1 instead? Training is especially important at level 1, where it may take the form of mentoring or of generic training in any of the skills of software engineering. The answer seems to be that *nothing* is placed at level 1, because level 1 is defined merely as not being at level 2. The hidden assumption here is that who we are, what problems we face, and what we're already doing doesn't matter: *just get to level 2*. In other words, the CMM doesn't perceive or adapt to the conditions of the client organization. Therefore training or any other informal practice at level 1, no matter how effective it is, could be squashed

accidentally by a blind and static CMM. Another example: Why is defect prevention a level 5 practice? We use project post mortems at Borland to analyze and improve our processes -- isn't that a form of defect prevention? There are many such examples I could cite, based on a reading of the CMM 1.1 document (although I did not review the voluminous Key Practices document) and the appendix of Humphrey's *Managing the Software Process [5]*. Basically, most and perhaps all of the key practices could be performed usefully at level 1, depending on the particular dynamics of the particular organization. Instead of actually modeling those process dynamics, the way Weinberg does in his work, the CMM merely stratifies them.

=B7 The CMM encourages displacement of goals from the true mission of improving process to the artificial mission of achieving a higher maturity level. I call this "level envy", and it generally has the effect of blinding an organization to the most effective use of its resources. The SEI itself recognizes this as a problem and has taken some steps to correct it. The problem is built in to the very structure of the model, however, and will be very hard to exorcise.

Feet of clay: The CMM's fundamental misunderstanding of level 1 Organizations

The world of technology thrives best when individuals are left alone to be different, creative, and disobedient. -- Don Valentine, Silicon Valley Venture Capitalist [8]

Apart from the concerns mentioned above, the most powerful argument against the CMM as an effective prescription for software processes is the many successful companies that, according the CMM, should not exist. This point is most easily made against the backdrop of the Silicon Valley.

Tom Peters's, *Thriving on Chaos* [9], amounts to a manifesto for Silicon Valley. It places innovation, non-linearity, ongoing revolution at the center of its world view. Here in the Valley, innovation reigns supreme, and it is from the vantage point of the innovator that the CMM seems most lost. Personal experience at Apple and Borland, and contact with many others in the decade I've spent here, support this view.

Proponents of the CMM commonly mistake its critics as being anti-process, and some of us are. But a lot of us, including me, are process specialists. We believe in the kinds of processes that support innovation. Our emphasis is on systematic problem-solving leadership to enable innovation, rather than mere process control to enable cookie-cutter solutions.

Innovation per se does not appear in the CMM at all, and it is only suggested by level 5. This is shocking, in that the most innovative firms in the software industry, (e.g., General Magic, a pioneer in personal digital communication technology) operate at level 1, according to the model. This includes Microsoft, too, and certainly Borland [2]. Yet, in terms of the CMM, these companies are considered no different than any failed startup or paralyzed steel company. By contrast, companies like IBM, which by all accounts has made a real mess of the Federal Aviation Administration's Advanced Automation Project,

score high in terms of maturity (according to a member of a government audit team with whom I spoke).

Now, the SEI argues that innovation is outside of its scope, and that the CMM merely establishes a framework within which innovation may more freely occur. According to the literature of innovation, however, nothing could be further from the truth. Preoccupied with predictability, the CMM is profoundly ignorant of the dynamics of innovation.

Such dynamics are documented in *Thriving on Chaos, Reengineering the Corporation* [4], and *The Fifth Discipline* [10], three well known books on business innovation. Where innovators advise companies to get flexible, the CMM advises them to get predictable. Where the innovators suggest pushing authority down in the organization, the CMM pushes it upward. Where the innovators recommend constant constructive innovation, the CMM mistakes it for chaos at level 1. Where the innovators depend on a trail of learning experiences, the CMM depends on a trail of paper.

Nowhere is the schism between these opposing world-views more apparent than on the matter of heroism. The SEI regards heroism as an unsustainable sacrifice on the part of particular individuals who have special gifts. It considers heroism the sole reason that level 1 companies succeed, when they succeed at all.
The heroism more commonly practiced in successful level 1 companies is something much less mystical. Our heroism means taking initiative to solve ambiguous problems. This does not mean burning people up and tossing them out, as the SEI claims. Heroism is a definable and teachable set of behaviors that enhance and honor creativity (as a unit of United Technologies Microelectronics Center has shown [3]). It is communication, and mutual respect. It means the selective deployment of processes, not according to management mandate, but according to the skills of the team.

Personal mastery is at the center of heroism, yet it too has no place in the CMM, except through the institution of a formal training program. Peter Senge [10], has this to say about mastery:

"There are obvious reasons why companies resist encouraging personal mastery. It is 'soft', based in part on unquantifiable concepts such as intuition and personal vision. No one will ever be able to measure to three decimal places how much personal mastery contributes to productivity and the bottom line. In a materialistic culture such as ours, it is difficult even to discuss some of the premises of personal mastery. 'Why do people even need to talk about this stuff?' someone may ask. 'Isn't it obvious? Don't we already know it?'"

This is, I believe, the heart of the problem, and the reason why CMM is dangerous to any company founded upon innovation. Because the CMM is distrustful of personal contributions, ignorant of the conditions needed to nurture non-linear ideas, and content to bury them beneath a constraining superstructure, achieving level 2 on the CMM scale may very well stamp out the only flame that lit the company to begin with.

I don't doubt that such companies become more predictable, in the way that life becomes predictable if we resolve never to leave our beds. I do doubt that such companies can succeed for long in a dynamic world if they work in their pajamas.

An alternative to CMM

If not the maturity model, then by what framework can we guide genuine process improvement?
Alternative frameworks can be found in generic form in *Thriving on Chaos*, which contains 45 "prescriptions", or *The Fifth Discipline*, which presents--not surprisingly--five disciplines. The prescriptions of *Thriving on Chaos* are embodied in an organizational tool called *The Excellence Audit*, and *The Fifth Discipline Fieldbook* [11], which provides additional guidance in creating learning organizations, is now available.

An advantage of these models is that they provide direction, without mandating a particular shape to the organization. They actually provide guidance in creating organizational change.

Specific to software engineering, I'm working on a process model at Borland that consists of a seven-dimensional framework for analyzing problems and identifying necessary processes. These dimensions are: business factors, market factors, project deliverables, four primary processes (commitment, planning, implementation, convergence), teams, project infrastructure, and milestones. The framework connects to a set of scaleable "process cycles". The process cycles are repeatable step by step recipes for performing certain common tasks.

The framework is essentially a situational repository of heuristics for conducting successful projects. It is meant to be a quick reference to aid experienced practitioners in deciding the best course of action.

The key to this model is that the process cycles are subordinated to the heuristic framework. The whole thing is an aid to judgment, *not* a prescription for institutional formalisms. The structure of the framework, as a set of two-dimensional grids, assists in process tailoring and asking "what if...?"

In terms of this model, maturity means recognizing problems (through the analysis of experience and use of metrics) and solving them (through selective definition and deployment of formal and informal processes), and that means developing judgment and cooperation within teams. Unlike the CMM, there is no a priori declaration either of the problems, or the solutions. That determination remains firmly in the hands of the team.
The disadvantage of this alternative model is that it's more complex, and therefore less marketable. There are no easy answers, and our progress cannot be plotted on the fingers of one hand. But we must resist the temptation to turn away from the unmeasurable and sometimes ineffable reality of software innovation.
After all, that would be immature.

Postscript 02/99

In the five years since I wrote this article, neither the CMM situation, nor my assessment of it, has changed much. The defense industry continues to support the CMM. Some commercial IT organizations follow it, many others don't. Software companies pursuing the great technological goldrush of our time, the Internet, are ignoring it in droves. Studies alleging that the CMM is valuable don't consider alternatives, and leave out critical data that would allow a full analysis of what's going on in companies that claim to have moved up in CMM levels and to have benefited for that reason.

One thing about my opinion has shifted. I've become more comfortable with the distinction between the CMM philosophy, and the CMM issue list. As a list of issues worth addressing in the course of software process improvement, the CMM is useful and benign. I would argue that it's incomplete and confusing in places, but that's no big deal. The problem begins when the CMM is adopted as a philosophy for good software engineering.

Still, it has become a lot clearer to me why the CMM philosophy is so much more popular than it deserves to be. It gives hope, and an illusion of control, to management. Faced with the depressing reality that software development success is contingent upon so many subtle and dynamic factors and judgments, the CMM provides a step by step plan to do *something* unsubtle and create *something* solid. The sad part is that this step-by-step plan usually becomes a substitute for genuine education in engineering management, and genuine process improvement.

Over the last few years, I've been through Jerry Weinberg's classes on management and change artistry: *Problem Solving Leadership*, and the *Change Shop*. I've become a part of his *Software Engineering Management Development Group* program, and the *SHAPE* forum. Information about all of these are available at http://www.ge=raldmweinberg.com. In my view, Jerry's work continues to offer an excellent alternative to the whole paradigm of the CMM: managers must first learn to see, hear, and think about human systems before they can hope to control them. Software projects are human systems=97deal with it.

One last plug. Add to your reading list *The Logic of Failure*, by Dietrich Dorner. Dorner analyzes how people cope with managing complex systems. Without mentioning software development or capability maturity, it's as eloquent an argument against CMM philosophy as you'll find.

References

1. Berti, Pat, "Four Pennsylvania schools await defense cuts.", Pittsburgh Business Times, Jan 22, 1990 v9 n24
2. Coplien, James, "Borland Software Craftsmanship: a New Look at Process, Quality and Productivity", Proceedings of the 5th Borland International Conference, 1994
3. Couger, J. Daniel; McIntyre, Scott C.; Higgins, Lexis F.; Snow, Terry A., "Using a bottom-up approach to creativity improvement in IS development.", Journal of Systems Management, Sept 1991 v42 n9 p23(6)

4. Hammer, Michael; Champy, James, Reengineering the Corporation, HarperCollins, 1993
5. Humphrey, Watts, Managing the Software Process, ch. 2, Addison-Wesley, 1989
6. Jones, Capers, Assessment & Control of Software Risks, Prentice-Hall, 1994
7. Paulk, Mark, et al, Capability Maturity Model 1.1 (CMU/SEI-93-TR-24)
8. Peters, Tom, The Tom Peters Seminar: Crazy Times Call for Crazy Organizations, Random House, 1994
9. Peters, Tom, Thriving on Chaos: Handbook for a Management Revolution, HarperCollins, 1987
10. Senge, Peter, The Fifth Discipline, Doubleday, 1990
11. Senge, Peter, The Fifth Discipline Fieldbook, Doubleday, 1994
12. Weinberg, Gerald M., Quality Software Management, v. 1 Systems Thinking, Dorset House, 1991
13. Weinberg, Gerald M., Quality Software Management, v. 2 First-order measurement, Dorset House, 1993