A pragmatic view of the Capability Maturity Model

Tim Mikkelsen, R&D Project Manager
TMO Software Technology Center
TN 679-3192
mailto:Tim_Mikkelsen@agilent.com

Executive Summary

The Capability Maturity Model (CMM) is a very useful tool. It is most effective when viewed and used as a means to a business objective and not an end in itself. The article describes some personal experiences, how to get started and challenges with CMM.

A brief review of the CMM

The Software Engineering Institute's (SEI) Capability Maturity Model (CMM) describes a framework that organizations can use to determine their ability to develop and maintain software. It is based on earlier process management work by Deming, Juran and Crosby but is specifically adapted to the needs of software projects and organizations.

CMM describes a continuum for software maturity ranging from ad hoc to mature and disciplined processes. It is built around the following, now familiar, 5 level model:

5	*Optimized*	Continuous process improvement
4	*Managed*	Product and process quality
3	*Defined*	Defined processes
2	*Repeatable*	Project management processes
1	*Initial*	Ad hoc or chaotic

There is a wealth of information available about SEI's CMM framework. A series of resources are listed at the end of the article.

Living through a CMM transition

I was involved with the transition of a moderate size software organization from level 1 (ad hoc) through level 2 (repeatable) up to level 3 (defined). The organization moved rather quickly through these phases (apparently faster than is normal). In general, the process helped the quality and effectiveness of the software development organization.

Prior to the use of CMM, we were a fairly typical HP software organization. Some of our processes were good, but many were very ad hoc and informal. We had the normal problems and associated project team heroics. With the transition to and use of the CMM we were able to deliver our defined products well. We really had a good handle on the software development process. We improved our software engineering professionalism. It did help and in several areas, the CMM made a big difference - especially in identifying risk areas in our development process.

A concern I had with the use of CMM was that development process became very 'turn-the-crank'. Note that the organization worked hard at having a flexible process and not become too rigid. In spite of the effort there was still an increase in bureaucracy. Some of this is a normal side effect and is really good in most of the software development process. However, the concern comes in that I felt that there was limited innovation and improvement in the product being delivered – even if it was on time with good quality. Related to my concern, there were also comments from some of the project managers that they felt like clerks filling out forms - rather than being paid for their judgement.

Is CMM worth it?

Out of this personal experience, I feel the CMM model is a very useful software management tool. But, it is crucial that you understand your project and organizational goals. A common failure is to attempt to 'achieve CMM level X'. The intent should be to improve software development to meet specific business goals. The goal should be of the form 'we need to improve aspect Y of our software development because it is necessary for our business goal Z'.

An example of a good goal driving CMM might be: "We need to improve our software quality because we are losing sales due to customer

dissatisfaction over product failures." Another example: "we need to reduce our product cycle time to more effectively compete with our major competitor." But the key is for everyone involved to internalize why you are doing CMM – that it is not just to get to an arbitrary level. CMM is just a tool – a very useful tool – but a tool that can be applied to meet a business need (or misapplied). The question of 'is it worth it' has the answer of 'yes, but only if it supports your business goals'.

The key to CMM, like any good software process, is that it is used as a means to achieve business goals as opposed to being an end in itself. A key corollary to this is that higher CMM levels are not necessarily better or appropriate – it depends on the goals and the character of the organization and products. If attaining a higher CMM level does not address a business need then it is probably not valuable. The common sense approach is:

1. Set the business goals
2. Understand what it takes to achieve the goals
3. Evaluate CMM as a possible tool
4. Determine which CMM level is appropriate

How do you get started?

Assuming that you do want to get started with CMM, the key question to ask is: "do I want to build CMM expertise inside my organization?" Generally speaking, if you are going to commit to a process, you need to build the internal knowledge. Given a desire to learn and implement CMM, a good set of steps include:

1. Identify a lead/champion for CMM
2. Get the lead trained and educated on CMM
3. Identify a trial project
4. Do an assessment of the project
5. Review the results
6. Train the managers and influencers on CMM
7. Train the organization on CMM
8. Do assessments of the organization
9. Review the results
10. Rinse and repeat (Plan/Do/Check/Act…)

This sounds a bit daunting - using the CMM framework does take time and effort. In my previous software R&D organization it took approximately ½ engineer per 10 developers to manage the CMM process. In addition to the process management, there is work that project managers and engineers need to do. However, if the process is thought-out carefully, the additional effort should be minimal or at least acceptable. And the desire is that you are not adding a new set of things on top of everything else you are doing, but that some of your current process efforts are replaced with more effective or focused efforts.

In terms of the training expense, there are a variety of studies that project training costs between $500 and $2000 per engineer. However, remember in all of this that organizations experience improvements in productivity, time to market, and post-release defects. (A study summarized return on investment to be approximately 5x.)

What are the barriers?

The CMM framework has spread fairly broadly. However, it is not yet universally accepted. Common concerns and issues include:

- the time to implement CMM
- the cost of implementing CMM programs
- the effort may be counter-productive
- the non-CMM issues are ignored
- the organization may become bureaucratic
- awareness of CMM
- lack of training on CMM
- cultural issues
- motivation
- and so on…

All of these barriers can be real problems, but in my view the organization motivation is the critical factor. Someone has to really drive for the use of CMM. This is the area where the tie to real business needs helps. If it is tied to real business needs that the manager, leader or champion can articulate, the rest of the organization can understand the need to adopt CMM. Otherwise, CMM can turn into 'yet another bureaucratic' process that the people in the organization have to deal with.

For organizations just getting started, there is another issue (besides motivation) that is critical - awareness. How do you know if you want to pursue CMM if you don't understand it yet? Although your business needs may indicate full commitment to the CMM, most organizations need to have a better sense of what they are getting into. Clearly, reading the literature and taking classes will help. Another very useful approach is to perform a CMM self-assessment. (There is a companion article on CMM self-assessment.)

Conclusions

The CMM software maturity model can be very useful. The key to CMM, like any good software process, is that it is used as a means to achieve

business goals as opposed to being an end in itself. The CMM process can be implemented with minimal or at least acceptable additional effort - when implemented in a careful and considered fashion. There can be a moderately healthy learning curve and cost associated with coming up the CMM - but the benefits can make it worthwhile. To quote one of my sources of inspiration: "Joe-Bob says check it out."

CMM Resources and Reading

1. "Software Process Improvement: 10 Traps to Avoid" by Karl Wiegers. Software Development, May 1996. Pages 51-58. A*n excellent, concise, article on the pragmatic aspects of software process.*
2. "Misconceptions of the CMM" by Karl Wiegers. Software Development, November 1996. Pages 57-64. *A quick sanity check on problems with the CMM.*
3. The Software Engineering Institute. URL: http://www.sei.cmu.edu/programs/sepm/process.html. H*ome page for the SEI's process technologies, of which the software CMM is only one. There are a variety of adaptations of the maturity model.*
4. The SEI's CMM. URL: http://www.sei.cmu.edu/cmm/cmms/cmms.html. *Home page for the Capability Maturity Model with links to a summary, related articles and how to obtain the official model.*
5. "Software Quality and the Capability Maturity Model" by J. Herbsleb, D. Zubrow, D. Goldenson, W. Hays and M. Paulk. – Communications of the ACM, June 1977, Volume 40, Number 6. Pages 31-40. *A good article on the CMM and its effectiveness.*
6. "Assessment Checklist for MTD CMM Process Assessment" by Scott Jordan. URL for Level 2: http://swtc.lvld.hp.com/~tim/swe/check-l2.html. URL for Level 3: http://swtc.lvld.hp.com/~tim/swe/check-l3.html.
7. "Software Process Profile: HP R&D Software Process Assessment" by Bert Laurence. HP's Software Initiative (SWI).
8. The Software Engineering Institute. URL: http://www.sei.cmu.edu. Phone: 412-268-5800. *Home page for the SEI organization.*
9. HP's Software Initiative. URL: http://www.sei.cmu.edu. Phone: 412-268-5800. *Home page for the SEI organization.*

BIOGRAPHY: Tim Mikkelsen started with HP in Loveland in 1977 working on instrument I/O. He has BS & MS degrees in CS (with EE minor) and recently received a MS in Management of Technology from NTU. Over the years Tim has been an R&D project and section manager, a cross-functional project and business team manager as well as having done rotations into marketing. He is co-author on a book on configuration management. Tim has a teenage son and daughter. He enjoys skiing, science fiction, and movies.