Ten Things Your Mother Never Told You About the Capability Maturity Model

Margaret Kulpa
Abacus Technology

This article discusses the 10 most common misconceptions the author has had to overcome concerning software process improvement and the Software Capability Maturity Model. Topics include management vs. developer changes required, having standards in place, consensus vs. steamroller approaches, keeping it simple, and why you cannot expect software process improvement to work unless you give your employees time to do it.

Most organizations that start out on the road toward software process improvement (SPI) using the Capability Maturity Model (CMM) for Software have no clue what this endeavor means. Most managers get sold on the idea based on competitive practices within the industry—"keeping up with the Joneses." This article discusses some common misconceptions about the torturous path to achieving a maturity level.

"What, me change? You've got to be kidding!"

Managers think the CMM focuses on changing the way the developers work. What happens is that the CMM forces management to change the way it manages projects. By requiring the development team to collect project data and report it to management, management becomes more aware of the project management process. In some organizations, managers do not want to know in detail what is really happening on their projects. The idea that someone would report to them actual schedule slippages and try to determine a standard deviation becomes incomprehensible. It is not uncommon to shoot the messenger.

What the CMM really provides is the ability to shape your own destiny. By generating procedures to do work, you control your work environment. If management understood that, they probably would not start CMM activities.

"We can't do this. We have to support our users."

It is amazing how often supporting the users is used as an excuse to not do CMM work. The CMM absolutely advocates supporting your users. That is why we are in this business. In case we have forgotten—no users, no work. Ultimately, by following CMM guidelines, supporting the user becomes easier because the ground rules have been established.

Change involves not only the developers but also management and the user community. No matter your position, your attitude plays an important role in SPI. For example, the way you do work in your twenties should be different from the way you do work in your forties, or at least it should be based on learning. If you are still doing things the way you always have, you need to reexamine your work and probably your life—and you are probably not the best person to be put in charge of the improvement effort. CMM work is all about change, something such people apparently know nothing about.

Users also need to change. Your users do not have the right to kill you, but that is what they are doing to our aging work force by creating unnecessary stress that contributes to heart attacks, cancers, and other ills. Control is the real issue. People who believe they have some control over their lives tend to be happier and live longer (so say the psychologists). So, to gain control of your project, you must control your users. Why should you accept an "emergency" request at 4 p.m. Friday that will keep you at work all night? Especially when it turns out that that particular user *always* turns in an "emergency" request at 4 p.m. on Friday and does not need the information until later the following week? Those users need to be trained in becoming pro-active and basically getting their act together. If everything is an emergency, nothing is an emergency. This sounds like an area in need of improvement.

"Standards? We don't need standards!"

Nowhere in the CMM does it say that standards are required. The CMM does not absolutely *require* anything. The model is not a step-by-step how-to model—it is a framework, a guideline. It tells you *what* you need to do but not *how* to do it. However, the CMM presupposes that you have standards and are trying to follow them. The standards they presuppose you already have are for products like coding standards, templates for a requirements specification, or test case scenarios.

Following standards institutes a basic structure within an organization. So, if you do not have any standards, get some. One place to search is Department of Defense (DoD) military standards, even if you are not a DoD organization. Start searching the Web for military standards as well as for the methods used to implement SPI. They are available, and they are free.

Just do not be anal when you interpret this information (see item 10, "Keep It Simple"). And all standards should be tailored for use in your organization. Do not think that you can use the same standards you used from the place you used to work in your new workplace. They do not fit. They cannot be used. They can be used as a target, but you will need to tailor them.

"Everybody knows what the process is. What's the big deal?"

Everybody knows what a process is until they try to define it in detail and write procedures that describe how to follow the process. Then, they shift back to documenting *who* needs to do something rather than on *how* that something is done. They also fall back on product standards (a form for documenting defects found during peer reviews) instead of process standards (*how* to perform the peer review, *how* to detect defects, and *how* to complete the form). Telling me that "it is the project manager's responsibility to determine schedule estimates" does not tell me *how* that manager is supposed to derive those estimates.

"Collaborative and achieving consensus ..."

CMM teams usually try to work collaboratively and make decisions by consensus. This concept is great and fosters buy-in and ownership but is extremely time-consuming and expensive. Consensus is *not* majority rules. Consensus means that everyone can live with the decision—they may not love it, but they can live with it. This way of working takes time. If you are on a tight schedule, (CMM work always is) you may need to stop the philosophizing and touchy-feely stuff and steamroll some folks. You will never get 100 percent buy-in from everyone. Take what you can get, and get those procedures written down.

"The CMM requires that a good process be in place."

No. It requires that a process be in place that is documented and followed. At first, your process could be awful. That is where the "continuous process improvement" concept comes in. After you hammer out a process, it is piloted, and projects start to use it, refinements will be made until (it is hoped) the process becomes "good." But to start, get something down on paper and use it. Clean it up as you go.

"We need to model our *as-is* process in order to create our *to-be* process."

Yes, but I find that organizations take up to a year to do this, only to find that their processes are too ad hoc to be used as a baseline of good practices and lessons learned. I suggest doing a software capability evaluation (which is now done for internal software process improvement) or a CMM-based Appraisal for Internal Process Improvement. These assessment methods can quickly determine consistent practices across the organization as well as strengths and weaknesses. Measurable action plans can be generated based on the results. Tracking progress can also be measured. The thing to remember before starting CMM activities is to determine ahead of time how to measure success. Modeling current processes is great—but will you ever see a return on that investment?

"Tie CMM activities to your business objectives."

Of course. There are some things in the CMM that may not make sense for you. For example, having a separate group to do software quality assurance (SQA) may not work if you only have 10 people in your company. The challenge is to figure out a way to perform quality assurance reviews and oversight in an objective, independent manner. And do not confuse "organization" with "company" or enterprise. An organization achieves a maturity level rating—not one project, not an entire company. Without going into detail, an organization *generally* consists of three to eight projects reporting to the same person, like a director or a division head—not an entire company (like IBM).

Do not get stupid about "business objectives." Ultimately, most organizations' business objectives are to achieve Level X by a certain date. If you are not currently doing SQA and do not want to do SQA (because of the cost and because it is overhead) yet you must achieve the level, do not try to be clever and tailor SQA out of the CMM process. Any certified evaluation team will catch you.

"Better, cheaper, faster."

This really irks me. When the CMM was written, most organizations had not yet begun the downsizing frenzy. Nowadays, however, organizations have cut their staff to the bare minimum. Management loves the maxim "better, cheaper, faster" and eventually, you will be able to turn out software of better quality, more quickly, and less expensively—*but not at first!* The average time to obtain your return on investment is three to five years.

SPI is expensive. Most organizations either hire outside consultants to start the journey or build it from the inside. Even if you are not hiring consultants, taking people away from coding, i.e., "real work," and having them do SPI costs you time, money, and schedule slippage. So management instead assigns SPI work in addition to existing work to an organization with extreme resource constraints, and it fails. You cannot squeeze additional effort from people who are already overworked. And having these people "work weekends, holidays, I don't care what it takes" violates the CMM principle of establishing and following *reasonable* plans.

"Keep it simple."

I like this one. Most organizations start off believing that they can keep their procedures simple—until they try to do it. Writing procedures that are simple and easy to follow, yet are thorough and complete, is extremely difficult. That is why the people on your teams need to be able to write and like to write as well as have a technical background and knowledge of the organization.

Managers in organizations today seem to feel that one person can wear many hats, i.e., a Powerbuilder programmer can also write procedures for how to write a requirements specification. Do you know what happens when you ask that unfortunate "techie" to do that? He breaks out in a cold sweat. Although some people are adaptable and can do many jobs, not everyone can do everything well. Different skillsets are required for different jobs.

Another problem is that teams often catch the improvement fever. They

want to improve everything. The challenge is to stay focused and use the CMM for software as your guide, but do not attack more than you can handle at one time. Remember: SPI is continuous improvement. It is iterative. Do what you can do in the time allotted, then go back and pick out more things once you have been allocated more time to do them.

Conclusion

Although there are other points to ponder when attempting this journey down the CMM path, these are the most frequently found errors made that I have documented. Good luck on your journey. ◆

About the Author

Margaret Kulpa is a consultant with Abacus Technology Corp. in Chevy Chase, Md. She is a certified lead evaluator and is authorized to teach the SEI's Introduction to CMM and the Software Capability Evaluation class. She has performed SPI duties for over 15 corporations and has evaluated over 30 organizations. She has also written and taught Key Process Area classes for Levels 2 and 3.

Abacus Technology
5454 Wisconsin Ave., Suite 1100
Chevy Chase, MD 20815
Voice: 301-951-1712
Fax: 301-907-8508
E-mail: kulpamk@songs.sce.com

METRICS, from page 26

be performed in an action plan allows the Measurement Team and manager to track progress with respect to the implementation of the measurement activities. An outline for an action plan follows:

1.0 Objective.
2.0 Description.
2.1 Background.
2.2 Goals.
- Business Goals.
- Measurement Goals.
- The Goals of This Plan.

2.3 Scope.
2.4 Relationship to Other Software Process Improvement Efforts.
2.5 Relationship to Other Functional Activities.
3.0 Implementation.
3.1 Activities, Products, and Tasks.
3.2 Schedule.
3.3 Resources.
3.4 Responsibilities.
3.5 Measurement and Monitoring.
3.6 Assumptions.
3.7 Risk Management.
4.0 Sustained Operation.

As the measurement activities are being planned, be sure to consider how the quality and success of the measurement activities will be measured. Building the need to measure the quality and success of the measurement activities into the measurement processes will help keep the activities aligned with the needs of the organization and mitigate some of the more common reasons why measurement fails. These reasons include lack of use of the data, personnel not understanding why the data need to be collected, and measurement viewed as an expendable, overhead activity. Following the goal-driven process outlined above provides a means to involve stakeholders, create understanding, and make measurement a part of the way the organization conducts business. Maintaining alignment between the measurement activities and the information needs of the organization helps the organization leverage information, which may otherwise not be captured, to enhance its performance. In summary, the goal-driven software measurement process directs attention toward measures of importance rather than measures that are merely convenient. ◆

About the Author

Dave Zubrow is team leader for software engineering measurement and analysis for the SEI and is assistant director of analytic studies for Carnegie Mellon University. He is an Assocation for Software Quality Certified Software Quality Engineer and a member of the Software Division Council for the American Society for Quality Control. He has a bachelor's degree from Penn State University and a master's degree and a doctorate from Carnegie Mellon University.

Voice: 412-268-5243
Fax : 412-268-5758
E-mail: dz@sei.cmu.edu

References

1. Schiemann, William and John Lingle, "Seven Greatest Myths of Measurement," *IEEE Engineering Management Review*, Spring 1998, pp. 114-116.
2. Park, R., W. Goethert, and W. Florac, *Goal-Driven Software Measurement*, (CMU/SEI 96-HB-002) Software Engineering Institute, Carnegie Mellon University, 1996.
3. PSM96, *Practical Software Measurement: A Guide to Objective Program Insight*, Washington, D.C., Joint Logistics Commanders, Joint Group on Systems Engineering, March 1996.
4. Basili, V. and D. Weiss, "A Methodology for Collecting Valid Software Engineering Data," IEEE Transactions on Software Engineering, Vol. 10, No. 6, 1984, pp. 728-738.
5. Briand, L., C.M. Differding, and H.D. Rombach, "Practical Guidelines for Measurement-Based Process Improvement," Software Process Improvement and Practices, Vol. 2, 1996, pp. 253-287.
6. Park, Robert E., et al., *Software Size Measurement: A Framework for Counting Source Statements* (CMU/SEI-92-TR-20), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., September 1992.
7. Florac, William A., et al., *Software Quality Measurement: A Framework for Counting Problems and Defects* (CMU/SEI-92-TR-22), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., September 1992.
8. Goethert, W., et al., *Software Effort Measurement: A Framework for Counting Staff-Hours* (CMU/SEI-92-TR-21), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., September 1992.