Essence of the Capability Maturity Model

Judy Bamberger, Process Solutions

As a student of software development dynamics, I'm annoyed by the CMM. I think it profoundly oversimplifies the software process problem—offering little of use to those who already understand software development and dangerous advice to those who don't. That's why I'm fascinated by Judy Bamberger. Judy is no mere process bureaucrat—she is a passionate and original thinker. Yet she applies the CMM. Moreover, she tames it. She makes it jump through hoops of fire. She brings to the CMM wisdom that is nowhere present in its text, and ignores any of its text that offers no wisdom. I sense a lot of reality in her approach and I'd like to understand it better.

—James Bach

Not too long ago, I was chatting with a vice president of a major shrinkwrapped software vendor. He described a new concept the firm had invented, something that was working extremely well. I think he called it a bug council: A regular meeting where requests for changes and remaining known defects were reviewed; their impacts discussed; decisions made on what to add, fix, or leave; and cross-functional commitments made to take action.

I smiled and asked, "What if I had told you that we've been using the same process you just 'discovered' for 15 years, except we call it a configuration control board. How would you have reacted?"

"Oh," he said, "I probably would have just ignored you!"

I asked again, eager to learn and ready to offer assistance, "How could I share any of my 15 years of experience with you, so your organization could learn from the rest of industry and gain benefit faster than having to invent these things for yourselves?"

The response was a weak "I don't know" and a blank stare.

And so finally I asked, "Are you aware of the Capability Maturity Model for Software or other software development guidelines that might provide some additional perspectives?"

"Yes," he said, "but none of that stuff applies to us; we are very different."

CMM IN CONTEXT

I was a key author of the CMM, contributing heavily to V1.0 and participating in all reviews. Since the release of CMM 1.0 in August 1991, I have had to deal with an amazingly varied set of misconceptions about the CMM. I am finding that the CMM is one of the most misunderstood pieces of technical literature in existence.

The CMM (as well as numerous IEEE standards and guidelines) attempts to collect some of the best wisdom in a limited portion of the software development industry. However, because of its size and dense presentation, I find people are having a difficult time grasping the CMM.

I agree with many of you that the CMM reflects the large, aerospace, contract-software development environment it was originally intended to address. I agree with many of you that the CMM touches on only a small part of good practices for software development (domain, technical, systems, and "people stuff" are not covered). And I agree with many of you that the language is intractable (there is English and then there is "CMM-speak").

These are some of the challenges I face as I work with organizations that have chosen the CMM as one tool to improve their development processes.

What I have learned is just how easy it is to perceive all kinds of things about the CMM that were not intended—just how easy it was to make a radically different *meaning* (interpretation) out of the bare text offered as *intake*. I have also learned just how deeply some of these *meanings* touch the emotions of those reading the CMM—the *significance* given to the *intake* and the *meaning* was something I had not imagined when I was writing and reviewing many of the CMM words that still exist today.

The CMM touches on only a small part of good practices for software development.

This leads to a puzzle for me. If the CMM really was written to reflect the large, aerospace, contract-software development environment, what is my responsibility as an author for translating the CMM and making it accessible to other people? What is my responsibility when I work with my clients, when I teach my students, when I talk with the community at large? What is the responsibility of those who read the CMM who are not from that original community? What is the responsibility of the "owners" of the CMM in all this?

I have concluded that the single part of the puzzle over which I have any control is how *I* deal with the CMM. So I always attempt to understand where the client (or student) is coming from; the problem or frame of reference she brings. I try to demystify the CMM, to bring it into her world using her vocabulary. I am told that I am very successful in doing this, because I demonstrate that the client (or student) is in charge of what she can or will understand.

When I work with my clients, we look for the essence of the CMM, the benefits it may bring to them. To this end, I often

ignore the concept of maturity levels and ratings, as we often find them more harmful than useful. What remains are many bits and pieces of the CMM that my clients find extremely beneficial, as they seem to address real problems my clients are having.

REPEATABLE LEVEL: STABILIZING THE PROJECT

I hear from many teams, projects, and organizations who believe their software development is "out of control." These are the issues my clients say cause them the most grief:

- not knowing which version of each file is "official";
- one person's fixes wiping out another's;
- features in one version not being retained in a later version;
- a difference in the marketing group's view of the features to be delivered and the development group's view (often with no real understanding of the end user's needs);
- features that change until the last moment and are requested in a seemingly ad hoc manner;
- not being able to meet development commitments and not being able to "push back" when developers believe the required commitments cannot be met;
- lack of insight as to how much more work there is to complete in order to meet the commitments;
- significant difference among the work products created by each developer, resulting in much confusion during integration and rework;
- an increasing amount of time spent in rework versus new development.

And there are the emotional results: frustration, demotivation, ambivalence, burnout, and fear (of being honest; of delivering late).

I start by trying to understand the issues that the people are facing. I listen; I observe; I reflect back; I get confirmation. I hear their issues—the facts and the emotions. And this set of concerns is what I hear and see in many places in the software industry.

This is what the CMM addresses in the Repeatable Level: getting some basics under control so that developers can do what they were hired to do—develop software. This is the essence of the CMM at the Repeatable Level:

- Get control of the product being released, and get control of the product pieces under development (the CMM's Software Configuration Management key process area).
- Define the feature set, ensure that what users want is what marketing and development think they want, and control the feature set so that the impact of changes made are more fully understood (the Requirements Management key process area).
- Use the feature set for estimating the work (to ensure that all promised features will be provided), and use the estimates for creating the schedule and other plans, leveraging as much past experience and knowledge as possible (the Software Project Planning key process area).
- Make the schedules and plans visible, so everyone knows their targets and what will happen if they cannot meet them; track progress, to celebrate success and to understand better the impacts of schedule slips or feature changes; keep managers informed of status, risks, and problems, so they can help; modify schedules and plans when the basic assumptions change (the Software Project Tracking and Oversight key process area).
- Help the development team state clearly what it plans to do and the standards and conventions it wants to adopt and follow; ensure the team follows its plans, standards, and conventions; and identify and correct discrepancies where they occur (the Software Quality Assurance key process area).

All these things taken together help establish some basic stability and visibility—for the developers, for the managers, for the testers, for the marketing group, for everybody. By making these things visible, everyone involved, directly or indirectly, has a much better chance of succeeding, having fun, and spending more time doing value-enhancing activities.

And what is really valuable to me is that these concepts apply to me, an individual, as well, whether I am writing code or developing training or creating a proposal or doing most any other consulting activity.

COMMON FEATURES: NOTHING IS MAGIC

I have never seen the essence of the Repeatable Level appear magically, even when a team, project, or organization recognizes it can gain substantial benefit from doing Repeatable Level activities. Here again, I ask questions and listen, observe, reflect, and confirm. I ask questions like:

What the CMM Is and What It Isn't

As an author of the CMM, I believe that it was not designed nor intended to be used in a way that says you must achieve level x by year y or that you have to "be" a certain level before people will do business with you. Nor do I believe that there is only one way and that is the CMM, nor any of the other slogans and requirements that seem to be increasingly prevalent.

The CMM wasn't intended to be all things to all people or cover all possible aspects of software and system development. The view that guided me during my many years' work as a CMM author, contributor, and reviewer was that the CMM was intended to provide one set of guidelines for managing software development projects and making improvements over time. This set of guidelines was based on best practices, software engineering discipline, real-world experience, and extrapolation from other industries. And, most importantly, this set of guidelines was just that—guidelines—not requirements or a checklist of "must do" items; the guidelines were intended to be interpreted, tailored, and applied within the culture and context of each unique organization.

For more information about the CMM, see http://www.sei.cmu.edu/products/.

- How do you *enable* good practices to happen? and
- How do you *evaluate* that you are doing those practices?

And this is the essence of the common features of the CMM—those *enablers* and *evaluators* that help ensure that what people want to do is, in fact, being done and is being done right the first time and every time. A good set of enabling and evaluating practices helps developers know they are doing things correctly and well and that the resulting products are achieving full value from them.

Enablers

To my questions about enablers (How do you capture and share good practices? How do new people come up to speed with them?), I often get answers like: We assign someone responsibility. We get training and tools. We make a checklist or create a template.

> What is really valuable about the CMM is that its concepts apply to me and my business.

Enablers also include policies and procedures. Now, sometimes when I talk about policies and procedures, people's eyes glaze over, as they think about five-inch binders (invariably covered by five inches of dust!). But what I mean by policies and procedures is whatever makes sense, given the size, experience, and stability of the organization.

I often tell the true story of the "yellow-stickie configuration management policy and procedure." Several years ago, when there were only 8-bit PCs, when there was no such thing as Windows, and when there were no configuration management tools for the PC environment, three very bright people were developing (quite rapidly) a large software product on four different systems.

Because they were very bright, they realized just how easily they could obliterate each other's work. So they defined and followed this protocol: "If you want to change a file that has been baselined, you must go to the machine maintaining the baseline. If you do not see a yellow stickie with the name of that file on it, you may check it out, but only after you post a yellow stickie with the name of that file on it. After you return the file, remove the yellow stickie. No file may be returned until it has been tested in your environment. No interfaces or common files may change without prior coordination among all of us."

In the CMM, enablers include things like checklists, templates, training, tools, funding, policies, procedures—all those items that provide people with the knowledge, skills, and tools to do the job right the first time and every time. In the terms of the CMM, these are the Commitment to Perform and the Ability to Perform common features.

Evaluators

To my questions about evaluators (How do you know where you are? How do you know that your practices are working well for you? How do you know that people have enough information to make the decisions they need to make?), I often get answers like: We check off a milestone. We hold a review. We provide a status report or other information to the project manager. We post progress or quality metrics so all can see.

When I talk about metrics, people sometimes become fearful, because they fear the measures are going to be used against them. So I work with them to define metrics that make sense, and that help them know if they are achieving what they want. Three questions I use to help elicit metrics relevant to my client are:

- Where are we now?
- Where do we want to be?
- How will we know when we get there?

My clients and I brainstorm about metrics to answer these questions. Then we negotiate a set of items the client wants to measure to help determine how well they are getting what they want.

In the CMM, evaluators include things like knowing whether or not something has been done, its status, its quality, its effectiveness; and use of that information by those who need to know it, such as developers, project managers, and senior managers, so they can take action in a timely and effective manner. In the terms of the CMM, these are the Measurement and Analysis and Verifying Implementation common features.

All these things, taken together, help build a culture and a memory of "how we do things here." They establish a basis for learning, for taking action in a timely and well-informed way, and for improvement over time.

And what is really valuable to me is that these concepts apply to me, an individual, as well, and to any activity that I am trying to make an effective part of the way I do business.

This is my view of the essence of the CMM, or at least a part of it. Now it's your turn. Try this:

1. Think of part of your software development process that is in some pain or ready for a tune-up.
2. Think of several options to address those parts.
3. Look for a key process area in the CMM that sounds close to some of those options.
4. Read the goals of the key process area and ask, "What do I get by having this goal?"
5. If there is a relationship between the goals and what you identified as options, explore deeper into the common features and the key practices.
6. Keep an open mind; see what you can learn; find the essence behind the CMM words, capture it in your frame of reference, and use it mindfully to help you address the issues you face.
7. And let me know what you find as *your* essence of the CMM. ❖

Judy Bamberger has 20 years' experience developing software, leading teams, doing process improvement, and teaching. An independent consultant, she also teaches an award-winning class that distills many quality tools to their essence and has the students apply them in the contexts of a real team, project, and organization. Contact her at bamberg@eaglet.rain.com.