The Impact of Evolving the Capability Maturity Model to Version 1.1

Mark C. Paulk and Suzanne M. Garcia, Software Engineering Institute

Abstract

The Capability Maturity Model for Software (CMM) has evolved significantly since publication of the software process maturity framework in 1987. The intent of CMM Version 1.0 was to formalize and elaborate the concepts of software process maturity, making them more accessible to the software community. The intent of the CMM Version 1.1 was to do a minor revision that would improve the consistency of the structure of the key practices, clarify the concepts, and provide consistent wording. These CMM Version 1.1 changes should make the CMM easier to use. This article discusses the expected impact of this evolution on existing software process improvement programs and summarizes the changes between CMM Versions 1.0 and 1.1 at a high level of abstraction.

Introduction: The Evolution of the CMM

In August 1986, the Software Engineering Institute (SEI), with assistance from the MITRE Corporation, began developing a process maturity framework that would help organizations improve their software process. This effort was initiated in response to a request to provide the federal government with a method to assess the capability of its software contractors. In June 1987, the SEI released a brief description of the process maturity framework [1] and in September 1987, a preliminary maturity questionnaire [2].

After five years of experience with the software process maturity framework and the preliminary version of the maturity questionnaire, the SEI evolved the software process maturity framework published in 1987 into a fully defined model [3], using knowledge acquired from software process assessments, software capability evaluations, and extensive feedback from both industry and government. This model, the CMM, helps organizations measure organizational software process maturity and establish process improvement programs. Version 1.0 of the CMM was released in September 1991 [3,7].

Based on feedback from the software community, the CMM was revised, and Version 1.1 was released in February 1993 [5,6]. This article discusses the expected impact of the changes on existing software process improvement programs; the changes between CMM Versions 1.0 and 1.1 are summarized in Appendix A for the reader interested in more detail. Since the intent of the CMM Version 1.1 was to do a minor revision that would improve the consistency of the structure of the key practices, clarify the concepts, and provide consistent wording, these changes should make the CMM easier to use.

The Implications of Change

Although the CMM Version 1.1 effort was intended to produce a minor revision, almost every practice in Version 1.0 was changed in some way. As the summary in Appendix A indicates, the overall impact of the changes was to clarify the meaning of the practices and make the wording more consistent, without substantively changing their content or intent.

Many changes made implementing a key practice easier. For example,

- Distinguishing between training at Level 2 and required training at Levels 3 and above.
- Changing the training subpractices to examples.
- Changing *reviews and approves* to *reviews*.
- Changing the term *independently* in Software Quality Assurance (SQA) Goal 2 to *objectively*.
- Making the concepts and practices in Organizational Process Definition less specific to a particular process definition method.
- Combining practices in Software Quality Management (SQM) to reduce the number of points in the lifecycle for identifying product quality goals.

- Changing *requires* to *typically specifies*.
- Changing the measurement subpractices to examples.

Significant Differences Between Versions 1.0 and 1.1

There were three changes between Versions 1.0 and 1.1 that could have a noticeable impact on a software process improvement program.

1. **The removal of subjective wording from the goals was desirable given the proposed usage of the goals in rating satisfaction of the key process areas.**
   Moving subjective wording (such as realistic plans and other goodness attributes) to the introductory paragraphs of the key process areas could, however, mask the ultimate reason for doing CMM-based improvement: to develop more effective and efficient processes. Organizations that emphasize getting a good score may lose sight of why the CMM was created.

2. **The addition of wording recommending written organizational policies could be a challenge for some organizations, where the development of official policies is an arduous and lengthy process.** This seems to be an issue of interpreting the CMM appropriately. A memo from senior management can set organizational expectations for performing a process, and setting expectations is the reason for establishing a policy. The same substantiation criteria would be applied in both cases for actually following the "policy." This ties back to the need for using professional judgment in interpreting the CMM and not using a checklist-oriented approach for software process improvement.

3. **The increase in consistency of wording of the key practices in general might lead organizations focusing on the key practices as their primary vehicle for improve-ment into believing the CMM has become more rigorous.**
   Again, the changes made that might lead to this perception reflect a recognition that Version 1.0 lacked consistency of wording, which could lead to inconsistent usage of the CMM.

Significant Differences Between the 1987 Maturity Questionnaire and the CMM

The changes with larger impact than those between the two versions of the CMM are those between the CMM and the 1987 maturity questionnaire, which was the starting point for most assessments and evaluations even after the CMM was released. The rating algorithm is a major difference between the two. Rating in the 1987 appraisal methods was based on a percentage of maturity questionnaire answers at each maturity level. In contrast, a CMM-based appraisal uses satisfaction of key process areas to determine the maturity level rating. The impact of differences between the CMM and the 1987 maturity questionnaire could be significant. The 1987 questions map to the practices in the CMM with a reasonable degree of fidelity, but some questions moved down in maturity level.

Two questions from Level 3 in the 1987 questionnaire map to Level 2 practices in the CMM:

- 2.4.6. Is a mechanism used for ensuring compliance with the software engineering standards?
- 2.4.19. Is a mechanism used for verifying that the samples examined by Software Quality Assurance are truly representative of the work performed?

Software Quality Assurance has been part of the software process maturity framework since 1987, although it was called *product assurance* then [1].

Five questions from Level 4 in the 1987 questionnaire map to Level 3 practices in the CMM:

- 2.1.12. Are internal design review standards applied?
- 2.1.13. Are code review standards applied?
- 2.3.1. Has a managed and controlled process database been established for process metrics data across all projects?
- 2.4.2. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
- 2.4.10. Is there a formal management process for determining if the prototyping of software functions is an

appropriate part of the design process?

The reviews referred to in questions 2.1.12 and 2.1.13 are peer reviews. Peer review standards are an intrinsic part of a well-defined process for performing peer reviews, as is indicated in Activity 2 of the Peer Reviews key process area. The process database was incorporated into the organizational assets in Activity 5 of Organization Process Definition. The mechanism for assessing the software process and implementing improvements is the process focus usually known as a software engineering process group, as described in Activity 1 of Organization Process Focus. The decision whether to prototype is an integral part of planning a coherent, integrated engineering process, as described in Activities 8 and 10 of Integrated Software Management and Activity 3 of Software Product Engineering.

Although none of the practices represented by these questions is trivial, the framework in which they reside existed in 1987. In moving from the 1987 description of the software process maturity framework to the CMM, however, areas that were only hinted at in 1987 have been significantly elaborated. Although the seeds were present [4], the following key process areas are in a sense new to the CMM:

- Software Subcontract Management.
- Training Program.
- Integrated Software Management.
- Intergroup Coordination.

Subcontract management was considered an aspect of project management in 1987, but feedback from appraisals indicated that it was a significant problem in many organizations and should be elevated in status. Training was always an integral part of the maturity framework; the need to formalize the organization's responsibility for training as part of institutionalizing process definition was established in 1988 [4]. Integrating organizational processes and project management was a side effect of defining key process areas to reside at a single maturity level; the Level 3 practices for planning and managing a software project justified the creation of a key process area. Coordinating the different engineering groups for a project was identified as an issue in appraisals; the thrust towards concurrent engineering and integrated product teams reflects this concern from a system-wide perspective.

For those organizations that based their improvement programs solely on changing "No" answers to "Yes" on the 1987 questionnaire, the evolution to the CMM poses a formidable challenge. However, this evolution focuses improvement efforts on the underlying processes and the model rather than the questionnaire, which makes the likelihood of successful improvement much greater.

Conclusion

The Capability Maturity Model represents a "common sense engineering" approach to software process improvement. While the CMM is not perfect, it does represent a broad consensus of the software community and is a useful tool for guiding software process improvement efforts. It has evolved since 1987 to the current sophisticated model (CMM Version 1.1), and it will continue to evolve in response to the needs of the software community and the ongoing changes in the software field.

The CMM is a powerful tool that can help software organizations improve their software processes and acquisition organizations select and manage software suppliers. Since it is only a tool, it must be intelligently used to help organizations address their specific business needs. The purpose of the CMM is to describe good management and engineering practices as structured by the maturity framework.

Judgment is necessary to use the CMM correctly and with insight. Intelligence, experience, and knowledge must shape an appropriate interpretation of the CMM in a specific environment. That interpretation should be based on the business needs and objectives of the organization and the projects.

Mark C. Paulk and Suzanne M. Garcia
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Voice: 412-268-5794 (Paulk) 412-268-7625 (Garcia)

Fax: 412-268-5758
Internet: mcp@sei.cmu.edu
smg@sei.cmu.edu

References

1. Humphrey, Watts S., *Characterizing the Software Process: A Maturity Framework*, Software Engineering Institute, CMU/SEI-87-TR-11, DTIC Number ADA182895, June 1987.
2. Humphrey, Watts S., and William L. Sweet, *A Method for Assessing the Software Engineering Capability of Contractors*, Software Engineering Institute, CMU/SEI-87-TR-23, DTIC Number ADA187320, September 1987.
3. Paulk, Mark C., Bill Curtis, Mary Beth Chrissis, et al., *Capability Maturity Model for Software*, Software Engineering Institute, CMU/SEI-91-TR- 24, DTIC Number ADA240603, August 1991.
4. Paulk, Mark C., "U.S. Quality Advances: The SEI's Capability Maturity Model," *Proceedings of the Third European Conference on Software Quality*, Madrid, Spain, Nov. 3-6, 1992.
5. Paulk, Mark C., Bill Curtis, Mary Beth Chrissis, and Charles V. Weber, *Capability Maturity Model for Software, Version 1.1*, Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403, February 1993.
6. Paulk, Mark C., Charles V. Weber, Suzanne M. Garcia, Mary Beth Chrissis, and Marilyn W. Bush, *Key Practices of the Capability Maturity Model, Version 1.1*, Software Engineering Institute, CMU/SEI-93-TR-25, DTIC Number ADA263432, February 1993.
7. Weber, Charles V., Mark C. Paulk, Cynthia J. Wise, and James V. Withey, *Key Practices of the Capability Maturity Model*, Software Engineering Institute, CMU/SEI-91-TR-25, DTIC Number ADA240604, August 1991.

Appendix A: A Summary of the Changes Between CMM Versions 1.0 and 1.1

General Changes

Name changes. Obvious changes include the name changes for several key process areas:

- *Quality Management* was changed to *Software Quality Management.*
- *Process Measurement and Analysis* was changed to *Quantitative Process Management.*
- *Technology Innovation* was changed to *Technology Change Management.*

The common feature *Monitoring Implementation* was changed to *Measurement and Analysis*. (The same content, however, is covered under CMM Version 1.1 as for Version 1.0.)

Wording templates. In general, the language of the key practices was cleaned up throughout the CMM. Wording templates were developed as part of the document design criteria and used to add consistency throughout the CMM and to help users of the CMM understand where similar concepts are intended. When a template is not used, the difference is (usually) purposeful to convey a difference in the concept unique to that key process area.

For example, one of the templates is that the SQA group reviews or audits the activities and work products for each key process area; in Software Quality Assurance, independent experts are called on to review the SQA group's activities. There is a similar variation for Training Program since organizational training activities are likely to be beyond the scope of the SQA group.

Eliminating redundancy. Throughout the CMM, redundant practices were eliminated. For example, the intent of Ability 4 in Integrated Software Management of Version 1.0 on orientation in the organization's standard software process is covered in Ability 4 of Organization Process Focus in Version 1.1. Another example is the intent of Activity 2 in Software Product Engineering of Version 1.0 (on active participation in control of the system requirements) is covered in Activities 1 and 3 of Requirements Management in Version 1.1.

Cross-references. Cross-referencing was significantly expanded throughout the CMM. Where appropriate, specific key practices are referred to rather than the key process area as a whole. An index was also added to Version 1.1.

Organizational structure. The overview section for interpreting the CMM [7] on organizational groups and roles was expanded to provide better descriptions of the roles and organizational concepts used in the CMM. Conversely, the "conceptual organization chart" in Version 1.0 was removed to emphasize that each organization should map the CMM

roles into their organization.

Reviews and approves. The use of the wording *reviews and approves* or *reviews and agrees to* was carefully considered so that only in places where agreement was appropriate were these terms retained. Otherwise, only the term reviews was used. Usually the reviews terminology was used where the software engineering group would not normally be expected to have the authority to provide approval over the item.

Software work products. In appropriate places, the term *software work product* was substituted *for software product*. *Software work product* is a general term that includes both nondeliverable and deliverable products; *software product* includes only those products deliverable to the customer.

Maintenance. Some wording was changed or enhanced to accommodate the inclusion of maintenance explicitly (for example, *software development and/or maintenance*), where there are specific differences in the way initial development versus maintenance activities are planned and performed.

Subjective wording. All of the CMM goals were rewritten to emphasize process end states rather than results, and subjective wording, e.g., *effective*, was generally removed. This change was made to support use of the goals as an integrating framework for rating the key process areas. Each key practice maps to one or more goals. Satisfying all the goals can be considered as satisfying the key process area.

Changes in the Common Features

Organizational policy. The wording template for policies changed from "the organization follows a written policy for X" in Version 1.0 to "the project follows a written organizational policy for X" in Version 1.1. This reflects the emphasis in many key process areas on project activities. It does imply that organizational policies are required, even at Level 2, where the scope of the policy was ambiguous before. This decision reflects the emphasis of the CMM on organizational improvement.

"Requires" versus "typically specifies."
For the policy practices, the preamble "this policy requires that" was changed to "this policy typically specifies that" to remove any checklist implications within the policy content recommendations. Similarly, for key practices where the activity is performed "according to a documented procedure," the preamble to the subpractices "this procedure requires that" was changed to "this procedure typically specifies that." The key practices describe the normal behaviors expected in an organization. They are not intended to be a definitive requirements specification for the process.

Training examples. The subpractices in the training-related Ability to Perform key practices were changed to example boxes to permit organizational flexibility in determining specific training needs. The CMM does not intend to specify a training curriculum for each key process area. An organization should address its specific training needs relating to the process.

Process, activity, and task. To lessen confusion between *process, activity,* and *task,* the term *task* was used where a defined unit of work with known entry and exit criteria was meant. *Activity* was only used where a more general term was appropriate. A process is usually composed of activities, tasks, or (sub)processes; it remains a very flexible term.

Identifying performer. Where both organization and project activities would be expected to occur, the language explicitly identifies which entity is intended to be the performer.

Managed and controlled. The term "placed under configuration management" was replaced with "managed and controlled" for items that are not considered traditional product configuration items such as plans, measurement documentation, or process descriptions. Managed and controlled implies version control and change control, but not the formality of full configuration management.

Measurement examples. The subpractices for the Measurement and Analysis common feature were changed to example boxes to permit organizational flexibility in determining the types of measurements that are meaningful to an organization or project. The software measurement field is young; *best practice* is controversial and uncertain. The concept of measurement, however, is a crucial building block towards applying an engineering discipline to the software process.

Changes at Level 2

"Process" at the Repeatable Level. *Process* was generally replaced at Level 2 by *activity* or *procedure* and used at the higher levels in the context of the organization's standard software process or the project's defined software process. At many places in the higher levels, "according to a documented procedure" was replaced with "according to the project's defined software process" to reflect the existence and use of a standard process above the repeatable level.

Training at the Repeatable Level. The phrase *receive training* is used at Level 2 rather than *receive required training*, which is used at the higher levels. This is intended to reflect the assignment of Training Program to Level 3. No forward referencing (reference to requirements for a higher key process area) is done in Version 1.1.

Requirements Management. The changes in Requirements Management reflect a sharper focus on the viewpoint of requirements management as seen from the perspective of software engineering, while recognizing that the development and revision of the system requirements allocated to software is typically the responsibility of a group external to the software engineering group, e.g., the systems engineering group. Activity 1 in Version 1.0, regarding the documentation of allocated requirements, was changed to Ability 2 in Version 1.1, to reflect this separation of responsibility, since the software engineering group is typically not responsible for documenting the system (allocated) requirements.

Software Subcontract Management. The introduction to Software Subcontract Management includes a discussion of the role of strategic business alliances in subcontracting. However, applying this key process area to joint venture and other partnering agreements still requires significant judgment since the focus of the practices is on a principal/subordinate relationship, not an "equal footing" relationship.

Software Quality Assurance. Activity 5 was reworded so that the SQA group audits *designated software work products* rather than *representative samples*. An SQA group can designate work products based on a sampling technique; the terminology is intended to be more generic than was the case in Version 1.0.

In Goal 2 for SQA, the term *independently* (verifies) was changed to *objectively* to recognize that the evolution of the SQA function may take different organizational forms as an organization matures.

Software Configuration Management. In Software Configuration Management, rather than using the hierarchy of configuration item, configuration component, and configuration unit, the phrase *configuration item/unit* is used. This is intended to reflect the ongoing evolution of the terminology in the standards related to software configuration management and maximize the flexibility of the implementer.

Changes at Level 3

Interrelationships between the organization key process areas. The interrelationships between Organization Process Focus, Organ-ization Process Definition, and Integrated Software Management were made more explicit via cross-referencing. Organization Process Focus coordinates creation of the organization's standard software process and related process assets, contents of which are described in Organization Process Definition. These assets are tailored by the projects to create a defined software process in Integrated Software Management.

Training Program. During the 1992 CMM Workshop, a refocusing of the Training Program key process area to Skills Building was proposed. This revamping was prototyped, but the prototype key process area was very controversial. The resolution was a set of revisions to recognize explicitly that alternate training vehicles, which differ from formal classroom training, may be an appropriate alternate implementation of this key process area.

Software Product Engineering. Activities 7 and 8 in Version 1.0 were combined to recognize that system and acceptance testing are not necessarily different activities, especially within non-DoD environments. Integration testing was added in Version 1.1 as Activity 6.

Changes at Level 4

Focusing Level 4. The names of both Level 4 key process areas were changed in Version 1.1. Quality Management was changed to Software Quality Management since the CMM is written from the software perspective. *Process Measurement and Analysis* was changed to *Quantitative Process Management* to emphasize the quantitative nature of

Level 4.

Quantitative versus statistical. The term *statistical control* was replaced by *quantitative control*. This is intended to include a wide variety of statistical techniques and encompass such things as software reliability engineering, as well as techniques from statistical process control.

Quantitative Process Management. Differentiation, both in definition and in usage, was made between the process capability of the organization's standard software process, and the process performance of the project's defined software process. This issue is discussed in detail in [6].

Software Quality Management. Activities 6, 7, 8, 9, and 10 in Version 1.0 were folded into Activity 3 in Version 1.1. These were key practices describing the types and activities related to software quality goals for work products of lifecycle stages such as requirements, design, and code. Leaving these as key practices overemphasized their importance in relation to overall goal setting and measurement. References to process quality, specifically in Activities 4, 12, 13, and 15, were generally removed, and their contents, where not redundant, were included in Quantitative Process Management to focus this key process area more clearly on product quality measurement.

Changes at Level 5

Integrating change at Level 5. The name of *Technology Innovation* in Version 1.0 was changed to *Technology Change Management*, and the integration between Technology Change Management and Process Change Management was made more explicit via cross-referencing.

*(**Editor's Note:** This work is sponsored by the U.S. Department of Defense.)*
