A History of the Capability Maturity Model for Software

MARK C. PAULK
Carnegie Mellon University

The Software Engineering Institute developed a five-level Capability Maturity Model for Software that described how software organizations transform their capability for building software by focusing on software process improvement. The model was initially published in 1987 as a software process maturity framework that briefly described five maturity levels. The model was formalized as the Software CMM® with a detailed description of recommended software engineering and management practices when published in 1991. Version 1.1 of the model was published in 1993, and work on Software CMM v2 was nearing completion when it was halted in 1997 in favor of the CMM Integration™ (CMMI) effort, which integrated systems engineering, software engineering, and integrated process and product development into a single model. The success of the Software CMM inspired a variety of maturity models and other standards. This article describes the history of the model and some of the major decisions made in its evolution.

Key words

Capability Maturity Model, CMM, CMMI, key process area, maturity level, process area, software process, staged representation

SQP References

The Software Quality Profile
vol. 1, issue 1
Watts Humphrey

INTRODUCTION

In August 1986, the Software Engineering Institute (SEI™) at Carnegie Mellon University, with assistance from the MITRE Corporation, began developing a process maturity framework that would help organizations improve their software processes (Humphrey 2002). This effort was initiated in response to a request to provide the federal government with a method for assessing the capability of their software contractors. In June 1987, the SEI released a brief description of the software process maturity framework (Humphrey 1987a) and, in September 1987, a preliminary maturity questionnaire (Humphrey 1987b).

Based on experience in using the software process maturity framework and the maturity questionnaire for diagnosing problems and improving processes, the SEI formalized the concepts as the Capability Maturity Model for Software (Software CMM). Version 1.0 of the model was published in 1991 (Paulk et al. 1991; Weber et al. 1991). Version 1.1 was released in 1993 (Paulk et al. 1993a; 1993b) and published as a book in 1995 (Paulk et al. 1995a). The Software CMM has now been retired in favor of the CMM Integration (CMMI) model, but it inspired many other standards and frameworks, including the People CMM (Curtis, Hefley, and Miller 1995), the Systems Engineering CMM (Bate et al. 1994), and the Systems Security Engineering CMM (Hefner 1997). It was also was one of the drivers in the development of ISO/IEC 15504 (Process Assessment), notably Part 7 on the assessment of organizational maturity (ISO 2008).

The Software CMM had an enormous impact on the software community. It is estimated that billions of dollars were spent

on CMM-based improvement (Emam and Goldenson 1999). Many case studies and research reports have been published on its impact on productivity, cycle time, and quality (Krasner 2001; Clark 2000; Harter, Krishnan, and Slaughter 2000), and its maturity levels have been embedded in parametric cost models such as COCOMO II (Boehm 2000). This article discusses the development of the Software CMM and the critical decisions made as it evolved; it is an update of a previous description of the evolution of the model (Paulk 1995c) that bridges to its successor, the CMMI for Development (Chrissis, Konrad, and Shrum 2006).

PRECURSORS TO THE SOFTWARE CMM

The basic premise underlying the SEI's work on software process maturity is that the quality of a software product is largely determined by the quality of the software development and maintenance processes used to build it. The staged structure of the software process maturity framework is based on total quality management (TQM) principles that have existed for nearly a century. The work of Frederick Taylor and Frank Gilbreth on "scientific management" and time-and-motion studies in the early 1900s eventually led to the new discipline of industrial engineering (Hays 1994). In the 1930s, Walter Shewhart, a physicist at AT&T Bell Laboratories, established the principles of statistical quality control (Shewhart 1931). These principles were further developed and successfully demonstrated in the work of such authorities as W. Edwards Deming (1986) and Joseph M. Juran (1988).

In recent years, the TQM concepts have been extended from manufacturing processes to service and engineering design processes. The software process can be defined as a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products. As an organization matures, the software process becomes better defined and more consistently implemented throughout the organization. This, in turn, leads to higher-quality software, increased productivity, less rework, and improved software project plans and management. This is an adaptation of the Deming chain reaction shown in Figure 1. Perhaps the most important point to note in this chain reaction is that process improvement and quality management should affect business objectives.

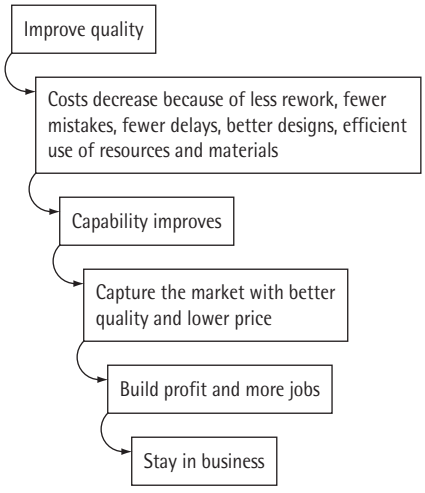
The Software CMM adapted TQM principles for software organizations. The model establishes a project management and engineering foundation for organizational learning and evidence-based management. Watts Humphrey adapted Philip Crosby's quality management maturity grid, described in *Quality Is Free* (Crosby 1979), for his process work at IBM (Humphrey 2002), and the maturity structure inspired the software process maturity framework that became the Software CMM.

Crosby's Quality Management Maturity Grid

Crosby describes five evolutionary stages in adopting quality practices. The quality management maturity grid applies five stages to six measurement categories in subjectively rating an organization's quality operation. The five stages are:

- *Uncertainty*: Management is confused and uncommitted regarding quality as a management tool.
- *Awakening*: Management is beginning to recognize that quality management can help.
- *Enlightenment*: The decision is made to really conduct a formal quality improvement program.
- *Wisdom*: The company has the chance to make changes permanent (things are basically quiet and people wonder why they used to have problems).
- *Certainty*: Quality management is considered an absolutely vital part of company management.

FIGURE 1 The Deming chain reaction



The six measurement categories are:

1) *Management understanding and attitude*: Characterized as "no comprehension of quality as a management tool" at uncertainty and "an essential part of the company system" at certainty.
2) *Quality organization status*. Characterized as hidden at uncertainty and a thought leader/ main concern at certainty.
3) *Problem handling*. Fought when they occur at uncertainty and prevented at certainty.
4) *Cost of quality as percent of sales*. Characterized as 20 percent at uncertainty and 2.5 percent at certainty.
5) *Quality improvement actions*. Characterized as no organized activities at uncertainty and a normal and continued activity at certainty.
6) *Summation of company quality posture*. Summarized as "we don't know why we have problems with quality" at uncertainty and "we know why we do not have problems with quality" at certainty.

Radice's IBM Work

Ron Radice and his colleagues at IBM, working under the direction of Watts Humphrey, identified 12 process stages, which were characterized by 11 attributes measured on a five-point scale (Radice et al. 1985). The process stages are stages in the life cycle: requirements, product-level design, component-level design, module-level design, code, unit test, functional verification test, product verification test, system verification test, package and release, early support program, and general availability. The 11 attributes include process, methods, adherence to practices, tools, change control, data gathering, data communication and use, goal setting, quality focus, customer focus, and technical awareness. The five-point scale consists of traditional, awareness, knowledge, skill and wisdom, and integrated management system. Humphrey brought these concepts to the SEI in 1986 and used them in defining the software process maturity framework.

THE 1987 SOFTWARE PROCESS MATURITY FRAMEWORK

In the software process maturity framework, Humphrey identified five maturity levels that described successive foundations for continuous process improvement and defined an ordinal scale for measuring the maturity of an organization's software processes (Humphrey 1987a; Humphrey 1988). The concepts underlying maturity levels have remained stable through the evolution of the Software CMM. In discussions of this early work, Bill Curtis, Humphrey's successor as director of the Process Program, identifies the focus on identifying and managing project commitments and on managing to a plan as the crucial differences between maturity models and Crosby's maturity grid. This operationally defined Level 2. It also reflects Beer, Eisenstat, and Spector's (1990) observation that senior managers create a climate for change in successful change programs, but this change needs to start at the grass-roots level rather than top-down.

The Five Maturity Levels

The maturity levels provide clear improvement priorities—guidance for selecting those few improvement activities that will be the most helpful if implemented immediately. This is important because most software organizations can only focus on a few process improvement activities at a time. Many organizations have made the mistake of identifying dozens of needed improvements, then failing to act on any of them because they were overwhelmed by the size and complexity of the need. The insight Humphrey provided in the maturity framework was an identification of the "vital few" issues at each maturity level that need to be tackled first. It was observed that challenged projects had many similarities in their underlying problems. Attacking common problems in a consistent manner was considered to be an effective way of building organizational capability, although it was recognized that specific projects would have unique needs that would also have to be addressed as part of an improvement program.

At Level 1, the *initial* level, the software process is characterized as *ad hoc*, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics. "Ad hoc" is sometimes used pejoratively, but ad hoc simply means "special." The problem with ad hoc processes is that it is difficult to predict performance or learn from experience when everything is new and unique. Level 1 is defined by the failure to satisfy the requirements for Level 2, yet there is one implicit requirement: If an organization does not build software, the model is irrelevant (and the organization could be characterized as a Level 0 organization).

At Level 2, the *repeatable* level, basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications. The focus at Level 2 does not explicitly include the engineering processes, because the major problems Level 1 organizations face are managerial problems, and not technical problems (DOD 1988). Engineering processes are planned and tracked at Level 2, but they are not described in detail—or even listed in most versions of the model.

At Level 3, the *defined* level, the software process for both management and engineering activities is documented, standardized, and integrated into a set of standard software processes for the organization. Projects use an approved, tailored version of the organization's set of standard software processes for developing and maintaining software. The engineering processes are first explicitly addressed at Level 3, but they must be implemented at Level 1 if the organization is building software, even if those engineering processes are ad hoc and inconsistently performed. The emphasis of Level 3, however, is on organizational learning via process definition and improvement.

At Level 4, the *managed* level, detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled. This implies statistical thinking (Britz et al. 1996) and evidence-based management (Pfeffer and Sutton 2006), although these terms were not used in the early formulations of the model. It also should be noted that measurement and analysis occurs at all levels of the model, although it comes to the forefront at Levels 4 and 5.

At Level 5, the *optimizing* level, continuous process improvement is enabled by feedback from the process and from piloting innovative ideas and technologies. Applying statistical thinking enables the organization to understand process capability and confirm when there are statistically and practically significant differences in performance as the process is changed. In the 1987 framework, this level was named the optimized level, but it was renamed in early 1988 to the optimizing level because the continuous process improvement journey never ends.

The critical concepts of organizational maturity therefore include a focus on organizational transformation to address common project problems, an emphasis on management rather than engineering processes, and the establishment of an improvement road map targeting the vital few issues blocking success.

Key Actions to Advance

In 1987 Humphrey described the five maturity levels in terms of the key actions needed to advance from one level to the next (Humphrey 1987a). These key actions were the first high-level expressions of what became the key process areas in the Software CMM. The key actions for moving from Level 1 to Level 2 were:

- *Project management:* The fundamental role of a project management system is to ensure effective control of commitments. For software, this starts with an understanding of the magnitude of the job to be done. In the absence of such an orderly plan, no commitment can be better than an educated guess.
- *Management oversight:* A suitably disciplined software development organization must have corporate oversight. The lack of management reviews typically results in uneven and generally inadequate implementation of the process, as well as frequent over commitments and cost surprises.
- *Product assurance:* A product assurance group is charged with assuring management that the software development work is actually done the way it is supposed to be done. To be effective, the assurance organization must have an independent reporting line to senior management and sufficient resources.
- *Change control:* Control of changes in software development is fundamental to business and financial control as well as to technical stability. To develop quality software on a predictable schedule, the requirements must be established and maintained with reasonable stability throughout the development cycle.

The key actions for moving from Level 2 to Level 3 were:

- *Process group*: Establish a process group. This is a technical group with exclusive focus on improving the software process. Until someone is given a full-time assignment to work on the process, little orderly progress can be made in improving it.

- *Process architecture:* Establish a software process architecture, which describes the technical and management activities required for proper execution of the development process. The architecture is a structural decomposition into tasks, and each task has entry criteria, functional descriptions, verification procedures, and exit criteria.
- *Software engineering methods:* If they are not already in place, introduce a family of software engineering methods and technologies. These include design and code inspections, formal design methods, library control systems, and comprehensive testing methods.

The key actions for moving from Level 3 to Level 4 were:

- *Process measurement:* Establish a minimum basic set of process measurements to identify the quality and cost parameters of each process step.
- *Process database:* Establish a process database with the resources to manage and maintain it.
- *Process analysis:* Provide sufficient process resources to analyze these data and advise project members on the data's meaning and use.
- *Product quality:* Assess the relative quality of each product, and inform management where quality targets are not being met.

The key actions for moving from Level 4 to Level 5 were:

- *Automated support:* Provide automatic support for gathering process data.
- *Process optimization:* Turn the management focus from the product to the process.

PRECURSORS TO THE SOFTWARE CMM

Two methods for diagnosing the problems facing software organizations using the software process maturity framework were developed: software process assessments (SPA) and software capability evaluations (SCE). SPAs were designed for self-assessment as part of an improvement program. Assessments depended primarily on interview data, with the big question being, "If there was one thing you could change about your organization, other than your boss or paycheck, what would it be?" (Paulk, Humphrey, and Pandelios 1992). SCEs were designed for customers to use in source selection and contract monitoring of software suppliers. Evaluations depended primarily on documents and artifacts that substantiated the consistent implementation of the process (Besselman et al. 1993). The SEI used the term "appraisal" to include assessments and evaluations. These three synonyms became process jargon to make subtle but useful distinctions (those who argued in favor of "assessment for process improvement" and "assessment for source selection" with "assessment" as the generic term lost the argument at the SEI, but ISO/IEC 15504 uses assessment with appropriate modifiers, while CMMI uses CMMI-based appraisal).

From the beginning, there was concern about the reliability and consistency of appraisals (Besselman 1992). Much of the knowledge and understanding of maturity concepts was embedded in Humphrey's mind. Although the concepts were gradually assimilated by the staff in the SEI's Process Program, there was a need for an explicit model of the maturity concepts to support its broad deployment.

The 1988 SCE Training

Because of the need for consistency in evaluating potential suppliers in source selection, the Contractor Software Engineering Capability Assessment (CSECA) project, later renamed the Software Capability Evaluation (SCE) project, under Jim Withey, was proactive in trying to formalize the maturity concepts.

The initial training for SCE teams was developed in early 1988. One module in the SCE team training described the key challenges in moving between levels. Some of the 1987 framework's key actions were split and others were combined. At Level 2, project planning was split out from project management. Management oversight, which focused on senior management's responsibilities for overseeing projects, was folded into project management. At Level 3, process architecture was expressed as standards because of the difficulty many people had in understanding what a software process architecture is. (This remains a challenge today; process architectures were an important discussion topic at the May 2008 PrIME workshop (PrIME).) Training, which was at best implicit in the 1987 framework, was added. Testing and reviews were explicitly emphasized as distinct from the more general software engineering methods. At Level 4, process database was folded into

process measurement. At Level 5, process optimization was split into problem prevention and problem analysis. Changing technology refocused automated support on the process issues involved with continual technology transition. Continuous process improvement implies continuous change in response to the changing environment.

While the maturity questionnaire, viewed at the time as the yardstick for measuring maturity, remained constant, the SCE training was intended to help teams understand and apply the maturity concepts consistently. Some issues were clarified during questionnaire administration, most notably that "technical reviews" in the questionnaire referred to what are now called "peer reviews." This was not a matter of model evolution, but a problem in establishing a common terminology (and one of the contributions of the Software CMM was in establishing a common terminology within the software industry in a number of instances).

Software Process Domains in 1988

The solution to reliability and consistency concerns was clearly neither training—which had an intrinsically limited audience—nor revising the maturity questionnaire, although that was an issue of concern as well. The concept of software process domains was investigated in 1988. The domains described maturity in terms of how software processes matured as an organization moved up the five levels. The domains, as of a July 1988 workshop, were planning, measurement and analysis, process definition, verification and validation, subcontract management, software management, technology insertion, software engineering methods, training, and selection and retention (of staff).

Domains provided a partial ordering of practices in an evolutionary path that was quite attractive. Three characteristics of the domain approach were, in hindsight, problematic. First, the domain descriptions were fairly short, approximately one to two pages apiece. Second, domains spanned maturity levels. Third, there were complex interdependencies between different domains—especially between practices at different maturity levels in different domains. Consensus was never reached on this approach, given the interdependencies among domains and the difficulty in precisely determining level or domain boundaries.

The Normative Model in 1989

In 1989 a representation of maturity in terms of a "normative" model was developed that applied stability factors and maturity indices to unit operations (while this model was named normative, the label was not very informative or useful; the Software CMM was a normative model itself). This resulted in a three-dimensional matrix whose cells defined the fundamental concepts to be implemented. (The continuous representation in ISO/IEC 15504 and CMMI is a two-dimensional matrix of processes and process capabilities.) Stability factors were entities such as resources, training, tools, plans, policies, and responsibility. Maturity indices were attributes such as existence, review, selection, metrics, analysis, and monitoring. The unit operations were activities such as staffing, committing, planning, tracking, executing, documenting, and verifying. While this approach led to some significant insights affecting the later development of the Software CMM, it was difficult to explain. The approach was deemed too artificial to be useful in a technology transition effort, although it was one of the inspirations for the continuous representation.

Managing the Software Process in 1989

Humphrey published *Managing the Software Process* in 1989 (Humphrey 1989). This seminal work has been extraordinarily successful in motivating the application of process management principles to the software process. The book is divided into chapters that cover the different "entities" that are the focus of each maturity level, but its structure is more useful for improving than assessing.

At Level 3, reviews was renamed to inspections because some interpreted reviews as being management reviews, rather than "peer reviews" such as Fagan-style inspections or structured walkthroughs. Software configuration management was split into two components: one focusing on control of the code at Level 2, and one focusing on control of all software products at Level 3 as an integral part of a defined software process. Training was not explicitly addressed, but the individual chapters emphasized training.

At Level 5, contracting for software explicitly addressed the customer-supplier relationship in a disciplined fashion, emphasizing the ideas of trust and competence. The customer-supplier relationship is an integral part of TQM. It is worth noting that the 1989 chapter on contracting for software, which explicitly

addresses the customer-supplier relationship, was not fully addressed in any of the later versions of the Software CMM. CMMI explicitly addressed acquisition in one of its later variants (and there is now a CMMI for Acquisition) (SEI 2007; Gallagher et al. 2009), but in many ways there are still insights to be gained from this chapter of Humphrey's book.

Software CMM v0.2 in 1990

Humphrey selected Bill Curtis as his successor as director of the SEI's Process Program in February 1991. As a member of the Process Program Advisory Board, Curtis advocated a formal model to express the organizational maturity concepts so briefly described in the software process maturity framework and maturity questionnaire. When he became the director, Curtis involved the entire Process Program in brainstorming, drafting, and reviewing the model.

A draft of the Software CMM, version 0.2, was distributed for review in June of 1990. It described the maturity levels in terms of key process areas, although there could be key practices at different maturity levels. At Level 2, a key process area on software subcontract management was split out of the project management areas as a result of feedback from the software community on the importance of managing software subcontractors. At Level 3, software requirements analysis and software design were added as key process areas. With software testing, this covered everything except coding that had been covered in software engineering methods. Software engineering interfaces on the interface of the software engineering group with other organizational entities was added as a result of analysis of findings from software process assessments. Software inspections were renamed peer reviews to provide more flexibility in implementing a range of peer reviews, from structured walkthroughs to inspections. The software engineering process group was renamed organizational process focus to provide more flexibility in implementing an appropriate mechanism for addressing process issues. At Level 5, automating the software process was renamed process improvement to address the gamut of process improvement activities that include automation as one vital aspect.

Software CMM v0.6 in 1991

In June of 1991, version 0.6 of the Software CMM was released for review. In earlier drafts of the Software CMM, key process areas spanned maturity levels, but they did not span all maturity levels. An issue was raised regarding how best to present problems in a key process area that differed from the bulk of the practices in that area. The decision was made with version 0.6 to redefine the key process areas as residing at a single maturity level. As a result of this, software project management (later renamed integrated software management) was added at Level 3 to address software project planning and management issues. This is the point in time when the staged architecture was first fully and formally defined.

If key process areas span levels, then a more complete picture of process capability is provided, but the "vital few" issues that dominate organizational improvement may be lost in the detail. The emphasis on organizational improvement in the Software CMM means that some processes are prioritized to be documented and improved before others. The continuous representation used in ISO/IEC 15504-2 and CMMI takes the opposite tack of providing a comprehensive profile of process capability. Both representations have their strengths and weaknesses (Paulk1996a; Paulk 1999), and CMMI includes both as equals (SEI00a; SEI00b).

At Level 2, a new key process area was added on software requirements management to address the issue of volatile requirements. This was handled by inference before, although change control of requirements was explicitly described in the 1987 framework. At Level 3, organization process definition was split out from organization process focus; these two key process areas were repeatedly split and combined in the iterative development of the model. When combined, which is desirable because they are tightly coupled, they make a large key process area, but they have separable concerns. Software requirements analysis, software design, and software testing were recombined into software product engineering (which also included coding). Many of the practices in the earlier versions of the engineering key process areas were considered too prescriptive. In making them generic, it became clear that they should be combined into a single key process area, as was originally done in 1987. It was also believed that there was a need to emphasize commonalty and integration of the software engineering processes, rather than potentially disjoint engineering processes. At Level 5, technology management and process change management were split out of process improvement. Technology management could be characterized as the innovation-driven, or revolutionary,

aspect of process improvement, where defect prevention was an evolutionary approach to process improvement.

SOFTWARE CMM v1.0 IN 1991

Software CMM v1.0, published in August 1991 as two technical reports, "Capability Maturity Model for Software" (Paulk et al. 1991) and "Key Practices for the Capability Maturity Model" (Weber et al. 1991), formalized the description of the maturity levels in terms of key process areas. Except for Level 1, each maturity level was decomposed into several key process areas that indicated the areas an organization should focus on to improve its software process. Key process areas identify the issues that must be addressed to achieve a maturity level. Although other issues affect process performance, the key process areas were identified because of their effectiveness in improving an organization's software capability. Each key process area contained two to four goals, which may be considered the requirements of the model. The key practices, subpractices, and supplemental information were informative material to aid in interpreting the model. At Level 2, software requirements management was renamed requirements management to clarify that the requirements that are being controlled are the customer's, not those derived from a software requirements analysis.

SOFTWARE CMM v1.1 IN 1993

Software CMM v1.1 was published in February 1993 as two technical reports, "Capability Maturity Model for Software, Version 1.1" (Paulk et al. 1993a) and "Key Practices for the Capability Maturity Model, Version 1.1" (Paulk et al. 1993b). This was a minor revision of Software CMM v1.0. Change requests that involved radical revisions, for example, adding new key process areas, were deferred to Software CMM v2, which was planned for 1997. There were comparatively few substantive changes.

There was extensive public participation in this revision. Nearly 1000 external reviewers in the CMM Correspondence Group had the opportunity to comment on the various drafts leading to Software CMM v1.1. The CMM Advisory Board, composed of government and industry advisers, helped the SEI review and reconcile conflicting requests.

Although the Software CMM v1.1 revision effort was intended to be a minor revision, almost every practice in Software CMM v1.0 was changed in some way. The overall impact of the changes was to clarify the meaning of the practices and make the wording of practices more consistent, without substantively changing their content or intent.

One interesting change was the removal of subjective wording from the goals. This was desirable given the use of the goals in rating satisfaction of the key process areas. Moving subjective wording (such as "reasonable plans" and other goodness attributes) to the purpose and 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. The problem of dysfunctional behavior driven by targeting a maturity level rather than improvement against business objectives remains a perennial problem for model-based improvement.

SOFTWARE CMM v2C

Software CMM v2 went through three major review cycles. A number of major changes had been proposed for consideration, and the SEI followed a public review process in considering these proposals (Paulk, Garcia, and Chrissis 1996b; 1996c; Paulk 1997).

Some change requests involved having key process areas span maturity levels and measuring process capability directly. As part of SEI participation in ISO/IEC 15504 (Process Assessment), alternatives incorporating the continuous representation were investigated (Paulk 1995b; Paulk 1999). The continuous perspective describes the evolution of processes and provides greater flexibility and finer granularity in rating processes. Both the staged and continuous perspectives have value, and they are conceptually compatible (Paulk 1996a). The decision was made to make the relationships between the two perspectives explicit but leave the staged architecture as the primary representation. While Software CMM v2 remained a staged model, it explicitly stated relationships to the continuous representation. A maturity level goal for each key process area captured institutionalization as expressed in the generic practices in the continuous architecture.

At Level 2, software subcontract management was expanded to include off-the-shelf and customer-supplied software as software supplier management.

At Level 3, risk management was proposed as a new key process area. A software risk management key process area was prototyped, but it was decided that a goal on risk management in integrated project management

was the best solution. This was a highly controversial discussion; everyone agreed on the importance of risk management, but they disagreed on whether it should be embedded in the project management key process areas or established as its own key process area.

At Level 4, research suggested a strong correlation between higher maturity levels and systematic reuse (or product lines) (Besselman and Rifkin 1995). A new key process area on organization software asset commonality was planned to address reuse, reengineering, aligning with strategic business goals, and similar business and domain-related issues. Organization process performance was split out of quantitative process management, which was renamed statistical process management.

CMM INTEGRATION (CMMI)

Software CMM v2 was planned for release at the end of 1997. Its release was halted by direction of the SEI's sponsor in the Office of the Undersecretary of Defense (Acquisition and Technology) in favor of work on the CMMI in October 1997. Continuing with Software CMM v2 would have diverted resources from CMMI. CMMI integrated systems engineering from EIA/IS-731 (Systems Engineering Capability Model), software engineering from Software CMM v2C, and integrated process and product development from the Integrated Product Development CMM v0.98 in a single model that incorporated both the staged and continuous representations (SEI 2000a; SEI 2000b). A CMMI Steering Group was established to govern the CMMI effort.

The process areas in CMMI apply equally to systems and software engineering, and there are discipline-specific amplifications for each as appropriate. Integrated process and product development (IPPD) is addressed by adding process areas, and in one case a goal to an existing process area.

In CMMI, measurement and analysis became a separate process area at Level 2, where in the Software CMM it had been a common feature in every key process area.

CMMI included a process area on risk management at Level 3, which had been folded into integrated project management in the Software CMM v2C as a goal. CMMI split out process areas on requirements development, technical solution, product integration, verification, and validation. It made peer reviews a goal in verification. Level 3 also added a process area on decision analysis and resolution. To support integrated process and product development, material was added to integrated project management, and new process areas were added on integrated teaming, integrated supplier management, and organizational environment for integration. Intergroup coordination was folded into integrated teaming. At Level 4, organization software asset commonality did not make it into CMMI.

CMMI has continued to evolve, and version 1.2 is now available (Chrissis, Konrad, and Shrum 2006), with work in progress on version 1.3. Discussing the evolution of CMMI is beyond the scope of this article, but it is worth noting that the original software/systems/IPPD/sourcing selection CMMI has become the CMMI for Development, and CMMI constellations for Acquisition (SEI 2007; Gallagher et al. 2009) and Services (SEI 2009) have been recently published.

REFLECTING ON KEY DECISIONS

A number of key decisions supported the broad adoption of the Software CMM. Probably none had more influence than the use of the model in source selection by the Department of Defense (DOD). The Software CMM is usually viewed as a model for large, custom software development organizations since that was the focus of the informative material in the model. The hierarchical structure, however, was intended to support its broad use by expressing the model requirements as abstract goals, while providing informative material to support consistent interpretation of the intent of those goals (and the use of subjective wording such as "reasonable plans" and "effective actions" in the purposes of the key process areas also encouraged a business-driven perspective). Practices in the Software CMM were never required, although CMMI has added the useful characterization that they are "expected" rather than just informative (and since data are collected at the practice level in appraisals, this characterization highlights the potential for dysfunctional behavior on the part of both organizations and appraisal teams). At least to some degree, this objective of broad use was met: When the Software CMM was retired in December 2005, about 13 percent of assessed organizations employed 25 or fewer professionals (and about one third employed 50 or fewer)—not large organizations, and more than 80 percent of the Software CMM users were commercial/in-house organizations—not government/military agencies or contractors (SEI 2006).

The hierarchical structure of the model also allowed a more granular rating than the simple pass/fail of standards in general (Paulk 2008). The improvement road map identified high-priority opportunities for improvement and provided a risk-oriented yardstick for comparing organizations. The five-level scale is still relatively crude; thus, it was recommended that supplier selection use a risk profile rather than simply a maturity level rating (Besselman 1992; Besselman et al. 1993), a recommendation that is now largely forgotten or perhaps superseded by the capability levels in the continuous representation of CMMI.

The concept of organizational maturity focused attention on the vital few issues that needed immediate attention in most organizations. Some objected to this prioritization, and the continuous representation developed for ISO/IEC 15504 (and based on work that preceded the maturity levels) and adapted in CMMI addresses some of this concern. Even for organizations using the continuous representation, however, it appears that the bulk of the improvement actions follow the road map laid out in the staged representation. Since assessments have always allowed for non-CMM findings and findings unique to the needs of the organization and/or projects, this may suggest a refinement rather than a difference in improvement priorities using these two different representations.

One drawback to the use of maturity levels, however, has been the dysfunctional behavior associated with organizations more concerned with assessment results than improving against business objectives. Perhaps this is an inevitable result of the model's use in source selection (Austin 1996). Despite early SEI attempts to encourage the use of risk profiles rather than maturity levels, and the performance of evaluations in a specific (potential) customer-supplier relationship, today organizations are expected to periodically assess themselves, and customers filter out suppliers with less than a maturity Level 3 rating.

As can be observed in the evolution of the key process areas over time, while there were additions for various reasons (integrated software management was added as part of formalizing the maturity levels as a staged model), most changes were driven by trying to clarify the intent and communicate more effectively, for example, splitting and combining key process areas. Humphrey's vision for transforming organizations remained surprisingly stable over the life of the Software CMM (CMMI has added new material because of its broader scope).

Many design decisions remain topics of discussion for CMMI, especially in the area of high maturity. Some of these topics include:

- Whether generic practices should have an associated generic process area
- How to reliably assess generic processes, such as decision analysis and resolution
- How to describe high-maturity concepts and practices in a way that is understandable and useful
- How to use a product context (for example, product lines, domain engineering, reuse) in statistical thinking and quantitative management
- How to describe a quality culture where broad participation in improvement and empowerment of the workers is a reality

Decisions such as these may be described as simply refinements and clarifications of what is already there, but ultimately CMMs are about communication, coordination, and management. Even when they invoke simple concepts, those concepts are frequently hard to do and subject to abuse.

CONCLUSION

Perhaps the most useful observation from this history is that continuous improvement applies to the best practice models such as the Software CMM, just as it does to the software process. Since the Software CMM has now been officially retired, one can write its history, but it inspired many descendants, including notably CMMI, ISO/IEC 15504 (Process Assessment), and the People CMM, which are continuing the journey.

Some of the useful ideas captured in the Software CMM, such as combining product knowledge via reuse and product lines with process measurement at Level 4, have fallen by the way, at least for now. Other thoughts, such as adding practices on process leadership from senior management at Level 2, have never been tried, although the idea of a "Level 2 SEPG" captures some of that concept. The journey of continuous improvement for models as well as processes continues.

Two themes can be observed in this history. First, there is a need for best practice frameworks that can act as the basis for reliable and consistent appraisals and effective process improvement programs. These

frameworks need a regular and consistent structure for their concepts to aid consistent interpretation and implementation. While their use can lead to dysfunctional behavior on the part of organizations that simply want a certificate to hang on the wall, models such as the Software CMM engage the community in a meaningful discussion of best practices.

Second, these frameworks need to be abstract and flexible for use in a wide range of environments. While specialized variants, such as the Systems Security Engineering CMM, can be quite useful for niche domains, the source model needs to be broadly applicable. This was not always fully achieved with the Software CMM; it was usually viewed as a model for large, custom software development organizations, since that was the genesis of the informative material in the model. Variants for small organizations, research environments, and so on, were continually being touted, but few survived. As many discovered, the goal-based structure of the Software CMM made it powerful and flexible in the hands of experienced professionals. Narrower variants rarely thrived, although those variants adding content or focusing on an entirely different domain could be useful.

In the end, all models such as the Software CMM struggle with balancing generality, detail, and usability. To a large degree the Software CMM appears to have walked that tightrope, as seen in its adoption throughout the world and its inspiration of other improvement frameworks.

REFERENCES

Austin, R. D. 1996. *Measuring and managing performance in organizations*. New York: Dorset House Publishing.

Bate, R., S. Garcia, J. Armitage, K. Cusick, R. Jones, D. Kuhn, I. Minnich, H. Pierson, T. Powell, and A. Reichner. 1994. A Systems Engineering Capability Maturity Model version 1.0. CMU/SEI-94-HB-04. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

Beer, M., R. A. Eisenstat, and B. Spector. 1990. Why change programs don't produce change. *Harvard Business Review* (November/December): 158-166.

Besselman, J. J. 1992. A collection of software capability evaluation (SCE) findings: Many lessons learned. In *Proceedings of the Eighth Annual National Joint Conference on Software Quality and Productivity*, Arlington, VA, March, 196-215.

Besselman, J. J., P. Byrnes, C. J. Lin, M. C. Paulk, and R. Puranik. 1993. Software capability evaluations: Experiences from the field. *SEI Technical Review '93*.

Besselman, J. J., and S. Rifkin. 1995. Exploiting the synergism between product line focus and software maturity. In *Proceedings of the 1995 Acquisition Research Symposium*, Washington, DC, 95-107.

Boehm, B. W., C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. J. Madachy, D. Reifer, and B. Steece. 2000. *Software cost estimation with COCOMO II*. Upper Saddle River, N.J.: Prentice Hall.

Britz, G., D. Emerling, L. Hare, R. Hoerl, and J. Shade. 1996. *Statistical thinking. A Special Publication of the ASQC Statistics Division* (Spring).

Chrissis, M. B., M. D. Konrad, and S. Shrum. 2006. *CMMI: Guidelines for process integration and product improvement, second edition*. Boston: Addison-Wesley.

Clark, B. K. 2000. Quantifying the effects of process improvement on effort. *IEEE Software* 17, no. 6 (November/December): 65-70.

Crosby, P. B. 1979. *Quality is free*. New York: McGraw-Hill.

Curtis, B., W. E. Hefley, and S. Miller. 1995. People Capability Maturity Model. CMU/SEI-95-MM-02. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

Deming, W. E. 1986. *Out of the crisis*. Cambridge, Mass.: MIT Center for Advanced Engineering Study.

DOD. 1988. Excerpts from Fall 1987 Report of the defense science board task force on military software. *ACM Ada Letters* (July/August): 35-46.

Emam, K., and D. R. Goldenson. 1999. An empirical review of software process assessments. NRC/ERB-1065 (NRC 43610). National Research Council Canada, Institute for Information Tech.

Gallagher, B. P., M. Phillips, K. J. Richter, and S. Shrum. 2009. *CMMI-ACQ: Guidelines for improving the acquisition of products and services*. Boston: Addison-Wesley Professional.

Harter, D. E., M. S. Krishnan, and S. A. Slaughter. 2000. Effects of process maturity on quality, cycle time, and effort in software product development. *Management Science* 46, no. 4 (April):451-466.

Hays, D. W. 1994. Quality improvement and its origin in scientific management. *Quality Progress* 27, no. 6 (May):89-90.

Hefner, R. 1997. Lessons learned with the systems security engineering capability maturity model. In *Proceedings of the 19th International Conference on Software Engineering*, Boston, May, 566-567.

Humphrey, W. S. 1987. Characterizing the software process: A maturity framework. CMU/SEI-87-TR-11. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

Humphrey, W. S., and W. L. Sweet. 1987b. A method for assessing the software engineering capability of contractors. Carnegie Mellon University, Software Engineering Institute, CMU/SEI-87-TR-23, September.

Humphrey, W. S. 1988. Characterizing the software process. *IEEE Software* 5, no. 2 (March):73-79.

Humphrey, W. S. 1989. *Managing the software process*. Reading, MA: Addison-Wesley.

Humphrey, W. S. 2002. Three process perspectives: Organizations, teams, and people. *Annals of Software Engineering* 4:39-72.

ISO. 2008. ISO/IEC 15504-7. Technology - Process assessment- Part 7: Assessment of organizational maturity. Geneva, Switzerland: International Organization for Standardization and International Electrotechnical Commission.

Juran, J. M. 1988. *Juran on planning for quality*. New York: Macmillan.

Krasner, H. 2001. Accumulating the body of evidence for the payoff of software process improvement – 1997. In *Software Process Improvement*, eds. R. B. Hunter and R. H. Thayer, 519-539. New York: IEEE Computer Society Press.

Paulk, M. C., B. Curtis, M. B. Chrissis, E. L. Averill, J. Bamberger, T. C. Kasse, M. D. Konrad, J. R. Perdue, C. V. Weber, and J. V. Withey. 1991. Capability Maturity Model for Software. CMU/SEI-91-TR-24. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

Paulk, M. C., W. S. Humphrey, and G. J. Pandelios. 1992. Software process assessments: Issues and lessons learned. In *Proceedings of ISQE92*, Juran Institute, March, 4B/41-58.

Paulk, M. C., B. Curtis, M. B. Chrissis, and C. V. Weber. 1993a. Capability Maturity Model for Software, version 1.1. CMU/SEI-93-TR-24. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

Paulk, M. C., C. V. Weber, S. M. Garcia, M. B. Chrissis, and M. W. Bush. 1993b. Key practices of the Capability Maturity Model, version 1.1. CMU/SEI-93-TR-25. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

Paulk, M. C., C. V. Weber, B. Curtis, and M. B. Chrissis. 1995a. *The capability maturity model: Guidelines for improving the software process.* Boston: Addison-Wesley.

Paulk, M. C., M. D. Konrad, and S. M. Garcia. 1995b. CMM versus SPICE architectures. *Software Process Newsletter.* IEEE Technical Committee on Software Engineering (Spring).

Paulk, M. C. 1995c. The evolution of the SEI's Capability Maturity Model for Software. *Software Process Improvement and Practice* (Spring): 3-15.

Paulk, M. C. 1996a. Process improvement and organizational capability: Generalizing the CMM. In *Proceedings of the ASQC's 50th Annual Quality Congress and Exposition*, Chicago, May, 92-97.

Paulk, M. C., S. M. Garcia, and M. B. Chrissis. 1996b. The continuing improvement of the CMM, version 2. *Fifth European Conference on Software Quality*, Dublin, September.

Paulk, M. C., S. M. Garcia, M. B. Chrissis, and W. Hayes. 1996c. SW-CMM v2: Feedback on proposed changes. *IEEE Software Process Newsletter* 7 (Fall): 5-10.

Paulk, M. C. 1997. The Capability Maturity Model for Software, version 2. *Software Technology Conference*, Salt Lake City, April, 247-302.

Paulk, M. C. 1999. Analyzing the conceptual relationship between ISO/IEC 15504 (Software Process Assessment) and the Capability Maturity Model for Software. In *Proceedings of the Ninth International Conference on Software Quality*, Cambridge, MA, October, 293-303.

Paulk, M. C. 2008. A taxonomy for improvement frameworks. *World Congress for Software Quality*, Bethesda, MD, 15-18 September.

Pfeffer, J., and R. I. Sutton. 2006. *Hard facts, dangerous half-truths, & total nonsense: Profiting from evidence-based management.* Boston: Harvard Business School Press.

Radice, R. A., J. T. Harding, P. E. Munnis, and R. W. Phillips. 1985. A programming process study. *IBM Systems Journal* 24, no. 2.

SEI. 2000a. CMMI Product Development Team. CMMI for systems engineering/software engineering/integrated product and process development, version 1.02, Staged Representation. CMU/SEI-2000-TR-030. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

SEI. 2000b. CMMI Product Development Team. CMMI for systems engineering/software engineering/integrated product and process development, version 1.02, Continuous Representation. CMU/SEI-2000-TR-031. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

SEI. 2006. Process maturity profile: Software CMM 2005 end-year update. Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

SEI. 2007. CMMI for acquisition, version 1.2. CMU/SEI-2007-TR-017. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

SEI. 2009. CMMI for services, version 1.2. CMU/SEI-2009-TR-001. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

Shewhart, W. A. 1931. *Economic control of quality of manufactured product.* New York: Van Nostrand.

Weber, C. V., M. C. Paulk, C. J. Wise, and J. V. Withey. 1991. Key practices of the Capability Maturity Model. CMU/SEI-91-TR-25. Pittsburgh: Carnegie Mellon University, Software Engineering Institute.

BIOGRAPHY

Mark Paulk is a senior systems scientist in the Institute for Software Research at Carnegie Mellon University in Pittsburgh. He researches and teaches on best practices for software engineering and service management. This includes empirical research and case studies of best practice, with an emphasis on high maturity, agile methods, measurement, and statistical thinking. From 1987 to 2002, he was with the Software Engineering Institute at Carnegie Mellon University, where he led the work on the Capability Maturity Model for Software. He was co-project editor of ISO/IEC 15504:2 (Process Assessment: Best Practices Guideline) from 1992-1995. Paulk received his doctorate in industrial engineering from the University of Pittsburgh, his master's degree in computer science from Vanderbilt University; and his bachelor's degree in mathematics and computer science from the University of Alabama in Huntsville. He is a senior member of the IEEE, a Senior member of ASQ, and an ASQ Certified Software Quality Engineer. His Web page is http://www.cs.cmu.edu/~mcp/ and his e-mail address is mcp@cs.cmu.edu.

Appendix

Evolution of the Software CMM

The following tables summarize the changes between different versions of the Software CMM as it evolved from a software process maturity framework in 1987 to Software CMM v1.1 and was replaced by CMMI. This comparison is only at the key process area level. Substantive changes occurred at the practice and subpractice levels, but the broad thrust of topics covered are reflected in this comparison.

The Evolution of Level 2

Software Process Maturity Framework	1988 SCE Training	Managing the Software Process	Software CMM v1.0	Software CMM v1.1	Software CMM v2c	CMMI/SE/SW/ IPPD v1.0
Change control of requirements			Requirements management	Requirements management	Requirements management	Requirements management
Project management	Project planning	The project plan	Software project planning	Software project planning	Software project planning	Project panning
Management oversight	Project management	Managing software organizations	Software project tracking and oversight	Software project tracking and oversight	Software project control	Project monitoring and control
			Software subcontract management	Software subcontract management	Software acquisition management	Supplier agreement management
Product assurance	Software quality assurance	Software quality assurance	Software quality assurance	Software quality assurance	Software quality assurance	Process and product quality assurance
Change control	Change control	Software configuration management I	Software configuration management	Software configuration management	Software configuration management	Configuration management
						Measurement and analysis

Appendix (continued)

The Evolution of Level 3

Software Process Maturity Framework	1988 SCE Training	Managing the Software Process	Software CMM v1.0	Software CMM v1.1	Software CMM v2c	CMMI/SE/SW/ IPPD v1.0
Process group	Process groups	The Software Engineering Process Group	Organization process focus	Organization process focus	Organization process focus	Organizational process focus
Process architecture	Standards	Defining the software process	Organization process definition	Organization process definition	Organization process definition	Organizational process definition
Software engineering methods		Software standards	Software product engineering	Software product engineering	Software product engineering	Requirements development technical solution product integration validation
	Testing	Software testing				
		Software configuration management II				
	Reviews	Software inspections	Peer reviews	Peer reviews	Peer reviews	Verification
			Intergroup coordination	Intergroup coordination	Project interface coordination	Integrated teaming
	Training	Addressed in the individual chapters	Training program	Training program	Organization training program	Organizational training program
			Integrated software management	Integrated software management	Integrated software management	Integrated project management (for IPPD)
						Risk management
						Decision analysis and resolution
						Integrated supplier management
						Organizational environment for integration

The Evolution of Level 4

Software Process Maturity Framework	**1988 SCE Training**	**Managing the Software Process**	**Software CMM v1.0**	**Software CMM v1.1**	**Software CMM v2c**	**CMMI/SE/SW/ IPPD v1.0**
Product quality	Quantitative quality plans	Managing software quality	Quality management	Software quality management	Statistical process management	Quantitative project management
Process measurement	Process measurement	Data gathering and analysis	Process measurement and analysis	Quantitative process management		
Process database						
Process analysis	Process analysis					
					Organization process performance	Organizational process performance
					Organization software asset commonality	

Appendix (continued)

The Evolution of Level 5

Software Process Maturity Framework	1988 SCE Training	Managing the Software Process	Software CMM v1.0	Software CMM v1.1	Software CMM v2c	CMMI/SE/SW/ IPPD v1.0
Process optimization	Problem prevention Problem analysis	Defect prevention	Defect prevention	Defect prevention	Defect prevention	Causal analysis and resolution
Automated support	Changing technology	Automating the software process	Technology innovation	Technology change management	Organization process and technology innovation	Organizational innovation and deployment
			Process change management	Process change management	Organization improvement deployment	
		Contracting for software				*Partially covered in the acquisition module of CMMI and the CMMI for Acquisition*