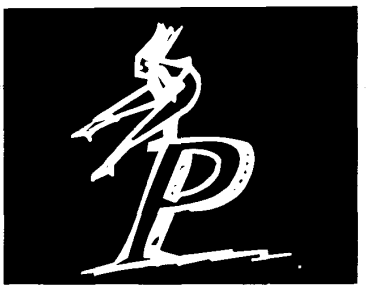


Capability Maturity Model, Version 1.1

MARK C. PAULK, BILL CURTIS, and MARY BETH CHRISSIS
Software Engineering Institute
CHARLES V. WEBER, *IBM Federal Systems Company*

◆ *The new version has more consistent wording and should be easier to use. It is based on more than six years of experience with software-process improvement and the contributions of hundreds of reviewers.*

After two decades of unfulfilled promises about productivity and quality gains from applying new software methodologies and technologies, developers are realizing that their fundamental problem is their inability to manage the software process. In many organizations, projects are often excessively late and over budget, and the benefits of better methods and tools cannot be realized in the maelstrom of an undisciplined, chaotic project.

In November 1986, the Software Engineering Institute, with assistance from Mitre Corp., began developing a process-maturity framework that would help developers improve their software process. In September 1987, the SEI released a brief description of the process-maturity framework[1] which was later expanded in Watts Humphrey's book, *Managing the Software Process*.[2]

The SEI also developed two methods — software-process assessment and software-capability evaluation — and a maturity questionnaire[3] to appraise software-process maturity.

After four years of experience with the process-maturity framework and the preliminary version of the maturity questionnaire, the SEI evolved the maturity framework into the Capability Maturity Model.

The CMM presents sets of recommended practices in a number of key process areas that have been shown to enhance software-development and maintenance capability. The CMM is based on knowledge acquired from software-process assessments and extensive feedback from both industry and government.

0740-7459/93/0700/0018/$03.00 © IEEE

The CMM guides developers on how to gain control of their development and maintenance processes and how to evolve toward a culture of software-engineering and management excellence. It was designed to help developers select process-improvement strategies by determining their current process maturity and identifying the most critical issues to improving their software quality and process.

By focusing on a limited set of activities and working aggressively to achieve them, a developer can steadily improve the organization-wide software process to enable continuous and lasting gains in capability.

The initial release of the CMM, version 1.0, was reviewed and used by the software community during 1991 and 1992. A workshop, held in April 1992, on CMM 1.0 was attended by about 200 software professionals. The current version of the CMM[4] is the result of the feedback from that workshop and ongoing feedback from the software community. In this article, we summarize the technical report that describes version 1.1; the box below summarizes the changes made.

IMMATURITY VERSUS MATURITY

Setting sensible goals for process improvement requires an understanding of the difference between immature and mature software organizations.

Immaturity. In an immature organization, software processes are generally improvised by practitioners and their managers during a project. Even if a software process has been specified, it is not rigorously followed or enforced.

The immature software organization is reactionary — its managers are usually focused on solving immediate crises (better known as fire fighting). Schedules and budgets are routinely exceeded because they are not based on realistic estimates. When hard deadlines are imposed, product functionality and quality are often compromised to meet a schedule.

An immature organization has no objective way to judge product quality or solve product or process problems. Therefore, product quality is difficult to predict. Activities intended to enhance quality, such as reviews and testing, are often curtailed or eliminated when projects fall behind schedule.

Maturity. A mature organization possesses an organization-wide ability to manage development and maintenance. Managers can accurately communicate the software process to staff and new employees, and work activities are carried out according to the planned process.

The mandated processes are usable and consistent with the way the work actually gets done. These defined processes are updated when necessary, with improvements developed through controlled pilot tests and cost-benefit analyses. Roles and responsibilities are clear within a project and across an organization.

In a mature organization, managers

SUMMARY OF DIFFERENCES BETWEEN CMM VERSION 1.0 AND VERSION 1.1

Most of the changes we made to CMM version 1.0 were done to improve the consistency of the key-practices structure, clarify concepts, and provide consistent wording. We made no changes to the high-level maturity framework. We also added an index and expanded and refined the glossary to provide additional clarification for terms.

We are working on a technical report that summarizes the changes and will include detailed traceability tables, summaries of the change for each key process area, and the document-design criteria. This is a summary of that draft document.

General changes. The names of some key process areas have changed, but the content of most areas remains the same.

We rewrote all the goals, to emphasize process end states rather than results, and to remove subjective words like "effective." The goals now serve as an integrating framework for rating key process areas: Each key practice maps to one or more goals, and each goal and its associated practices can be considered a subprocess area. Satisfying all the goals satisfies the key process area.

We also eliminated redundant practices and expanded cross-referencing significantly. When appropriate, we cross-reference directly to key practices rather than to the entire area.

Language changes. We developed and used wording templates to add consistency and to help users understand when we are and aren't talking about similar concepts — wording differences are now purposeful.

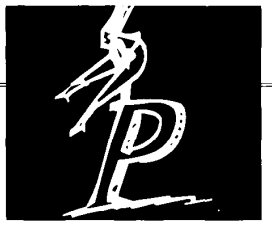
We expanded the overview section of the key practices that apply to groups and roles, to better explain these concepts. Conversely, we removed the conceptual organization chart, to emphasize that each organization should map the CMM roles to their own organization.

The wording template for policies changed from "the organization follows a written policy for X" to "the project follows a written organizational policy for X." 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. We made this change to reflect the CMM's emphasis on organizational improvement.

When both organizational and project activities are expected, the language in version 1.1 explicitly identifies the entity intended to be the performer.

When appropriate, we substituted the term "software work product" for "software product." These definitions are now generally consistent with IEEE usage, which defines a software work product to include both nondeliverable and deliverable products and a software product to include deliverables only.

We carefully considered the use of the wording "reviews and approves" or "reviews and agrees to;" only when the



monitor the quality of products and the process that produced them. There is an objective, quantitative basis for judging product quality and analyzing problems with the product and process. Schedules and budgets are based on historical performance and are realistic; expected results for cost, schedule, functionality, and quality are usually achieved.

In general, a disciplined process is consistently followed because all the participants understand the value of doing so, and an infrastructure exists to support the process.

FUNDAMENTAL CONCEPTS

A software process is a set of activities, methods, practices, and transformations that people use to develop and maintain software and associated products (project plans, design documents, code, test cases, user manuals, and so on). As an organization matures, the software process becomes better defined and more consistently implemented throughout the organization.

- *Software-process capability* describes the range of expected results that can be achieved by following a software process. An organization's software-process capability is one way to predict the most likely outcome of the next software project it undertakes.
- *Software-process performance* represents the actual results achieved by following a software process. Process performance focuses on achieved results; process capability focuses on expected results.
- *Software-process maturity* is the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective. Maturity implies a potential for growth in capability and indicates both the richness of an organization's software process and the consistency with which it is applied in projects throughout the organization.

As a software organization gains in maturity, it institutionalizes its software process via policies, standards, and organizational structures. Institutionalization entails building an infrastructure and a corporate culture that supports the methods, practices, and procedures of the business so that they endure after those who originally defined them have gone.

Five maturity levels. Continuous process improvement is based on many small, evolutionary steps rather than revolutionary innovations. The staged structure of the CMM is based on principles of product quality espoused by Walter Shewart, W. Edwards Deming, Joseph Juran, and Philip Crosby.

The CMM provides a framework for organizing these evolutionary steps into five *maturity levels*, shown in Figure 1, that lay successive foundations for continuous process improvement. The levels define an ordinal scale for measuring process maturity and evaluating process capability. The levels also help an organization prioritize its improvement efforts.

Each maturity level comprises a set of process goals that, when satisfied, stabilize an important component of a software

agreement aspect was appropriate did we retain that wording. Otherwise, we used only "reviews," usually when the software-engineering group would not be expected to have the authority to approve the item.

We replaced the phrase "This procedure/policy requires that:" to "This procedure/policy typically specifies that:" to remove the implication that what follows is a checklist. Key practices describe the normal behavior expected in an organization; they are not intended to be a requirements specification for the process.

To reduce confusion about the terms "process," "activity," and "task," we used "task" when we meant to describe a defined unit of work with known entry and exit criteria, and "activity" only when a more general term was appropriate.

Level 2 changes. In general, at this level we replaced the word "process" with "activity" or "procedure." In version 1.1, we reserve "process" for higher levels, in the context of an organization's standard or a project's defined process.

- *Requirements Management*: Here we tried to sharpen the focus on requirements management as seen from a software-engineering perspective, while recognizing that the development and revision of requirements typically is not the responsibility of the software-engineering group.
- *Software Project Planning*: We added a verification practice to address senior management's involvement in planning activities.
- *Software Project Tracking and Oversight*: Many of the changes in this area are intended to clarify who does what, emphasizing the software-engineering group's responsibilities.
- *Software Subcontract Management*: This area addresses the role of strategic business alliances in subcontracting. The focus of the practices is on a principal-subordinate, not an equal, relationship.
- *Software quality assurance*: We added ability 1 ("a group that is responsible for coordinating and implementing SQA for the project ... exists or is established") in the document-design criteria.
- *Software Configuration Management*: We replaced the hierarchy of configuration item, configuration component, and configuration unit with the phrase "configuration item/unit." This change reflects the ongoing evolution of the terminology in the standards world and maximizes flexibility.

Level 3 changes. We made the interrelationships among the Organization Process Focus, Organization Process Definition, and Integrated Software Management key process areas more explicit through cross-referencing.

- *Organization Process Focus*: This area covers the creation of the organization's standard process and related process assets.
- *Organization Process Definition*: This area describes the assets created above, which are

process. Achieving each level of maturity establishes a different component in a software process, resulting in an increase in the process capability of an organization. The labeled arrows in Figure 1 indicate the type of process capability being institutionalized by an organization at each step of the maturity framework.

Behavioral characterization. Maturity levels 2 through 5 can be characterized through the activities performed by an organization to establish or improve its software process, by activities performed on each project, and by the resulting process capability across projects. We include a behavioral characterization of level 1 to establish a base of comparison for process improvements at higher maturity levels.

♦ *Level 1: Initial.* At the Initial level, an organization typically does not provide a stable environment for developing and maintaining software. Such organizations frequently have difficulty making commitments that the staff can meet with an orderly engineering process, resulting in a

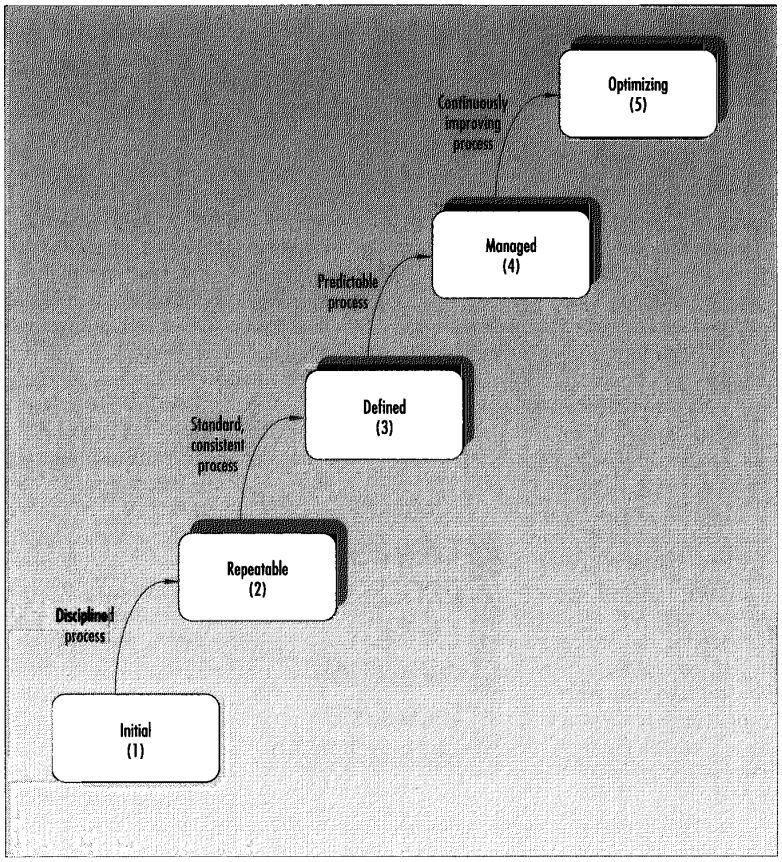


Figure 1. Maturity framework with five levels, each one the foundation for the next.

tailored by the projects to create a defined process in Integrated Software Management. We deleted activity 1, which described the process assets that must be developed and maintained as inappropriately prescriptive.

♦ *Training Program:* The 1992 CMM Workshop proposed refocusing this key process area to Skills Building, which was drafted but very controversial. This was resolved by recognizing explicitly that training vehicles other than formal classroom training may be an appropriate way to implement this key process area.

♦ *Integrated Software Management:* We removed redundancies with Software Project Planning and Software Project Tracking and Oversight.

♦ *Software Product Engineering:* We added ability 3 to address the concept of orientation and cross-training in disciplines other than an individual's main assignment.

♦ *Intergroup Coordination:* We changed the terminology about groups throughout, eliminating the term "project groups" in favor of "software-engineering group," "(other) engineering groups," "software-related groups," and "(other) affected groups," to clarify the interrelationships intended.

♦ *Peer Reviews:* We changed the specific data on the conduct and results of peer reviews in activity 3 to examples and dropped the organizational database.

Level 4 changes. We changed the names of both level 4 key process areas: Quality Management became Software Quality Management and Process Measurement and Analysis became Quantitative Process Management.

♦ *Quantitative Process Management:* We added commitment 1, which addresses policy implementation by the project. This key process area involves significant activities by both projects and the organization, although the majority of activities are project-focused.

♦ Software Quality Management: Activities 6 through 10 in version 1.0 were folded into activity 3 in version 1.1. These key practices described software-quality goals. Having them be key practices overemphasized their importance in relation to overall goal setting and measurement.

Level 5 changes. We changed the name of Technology Innovation to Technology Change Management and made the integration between Technology Change Management and Process Change Management more explicit, through cross-references.

♦ *Defect Prevention:* We made several wording changes throughout this key process area to differentiate between organization and project.

♦ *Technology Change Management:* We deleted subpractices for activity 7 because they were redundant to the subpractices of activity 5 in Process Change Management.

♦ *Process Change Management:* We moved ability 1, on establishing a software process improvement program, to become activity 1.

series of crises. During a crisis, projects typically abandon planned procedures and revert to coding and testing.

Success depends entirely on having an exceptional manager and a seasoned and effective development team. Occasionally, capable and forceful software managers can withstand the pressures to take shortcuts, but when they leave the project their stabilizing influence leaves with them. Even a strong engineering process cannot overcome the instability created by the absence of sound management practices. (Selecting, hiring, developing, and retaining competent people are significant issues for organizations at all levels of maturity, but they are largely outside the scope of the CMM.)

EVEN A STRONG PROCESS CANNOT OVERCOME UNSOUND MANAGEMENT.

In spite of this ad hoc, even chaotic, process, level 1 organizations frequently develop products that work, even though they may be over budget and behind schedule. Success in level 1 organizations depends on the competence and heroics of the people in an organization and cannot be repeated unless the same competent individuals are assigned to the next project. Thus, at level 1, capability is a characteristic of individuals, not organizations.

♦ *Level 2: Repeatable.* At the Repeatable level, policies for managing a software project and procedures to implement those policies are established. The planning and managment of new projects is based on experience with similar projects. Process capability is enhanced by imposing basic process-management discipline project by project.

Projects in level 2 organizations have installed basic management controls. Realistic project commitments are based on the results observed on previous projects and on the requirements of the current project. Project managers track costs, schedules, and functionality and identify problems in meeting commitments when they arise.

Software requirements and the work products developed to satisfy them are baselined, and their integrity is controlled. Project standards are defined, and the organization ensures they are faithfully followed. The project team works with their subcontractors, if any, to establish a customer-supplier relationship.

Processes may differ among projects in a level 2 organization. To achieve level 2, an organization must have policies that help project managers establish appropriate management processes.

The process capability of level 2 organizations can be summarized as disciplined because project planning and tracking are stable and earlier successes can be repeated.

♦ *Level 3: Defined.* At the Defined level, a typical process for developing and maintaining software across the organization is documented, including both software-engineering and management processes, and these processes are integrated into a coherent whole. The CMM calls this an organization's *standard* software process.

Processes established at level 3 are used (and changed, as appropriate) to help managers and staff perform more effectively. An organization exploits effective software-engineering practices when standardizing its processes.

A group — like a software-engineering process group[5] — is responsible for an organization's process activities. An organization-wide training program ensures that the staff and managers have the knowledge and skills they require.

Project teams tailor an organization's standard software process to develop their own *defined* process, which takes into account the project's unique characteristics. A defined process contains a coherent, integrated set of well-defined software-engineering and management processes.

A well-defined process includes readiness criteria, inputs, standards and procedures for performing the work, verification mechanisms (such as peer reviews), outputs, and completion criteria. Because the process is well-defined, management has good insight into technical progress on all projects.

The software-process capability of level 3 organizations can be summarized as standard and consistent because both software engineering and management activities are stable and repeatable. Within product lines, cost, schedule, and functionality are under control and quality is tracked. This process capability is based on a common, organization-wide understanding of the activities, roles, and responsibilities in a defined process.

♦ *Level 4: Managed.* At the Managed level, an organization sets quantitative quality goals for both products and processes and instruments processes with well-defined and consistent measurements. Productivity and quality are measured for important process activities across all projects as part of an organizational measurement program. Software products are of predictably high quality.

An organization-wide process database is used to collect and analyze the data available from a project's defined processes. These measurements establish the quantitative foundation for evaluating a project's processes and products. Projects control their products and processes by narrowing the variation in their performance to fall within acceptable quantitative boundaries. Meaningful variations in process performance can be distinguished from random variation (noise), particularly within established product lines. The risks involved in moving up the learning curve of a new application domain are known and carefully managed.

The software-process capability of level 4 organizations can be summarized as being quantifiable and predictable because the process is measured and operates within measurable limits. This level of capability lets an organization predict trends in process and product quality within the quantitative bounds of these limits. Because the process is both stable and measured, when some exceptional circumstance occurs, an organization can identify and address the *special* cause of the variation. When the known limits of the process are exceeded, managers take action to correct the situation.

♦ *Level 5: Optimizing.* At the Optimizing level, the entire organization is focused on continuous process improvement. The organization has the means to identify weaknesses and strengthen the process proactively, with the goal of preventing defects. Data on process effectiveness is used to perform cost-benefit analyses of new technologies and propose changes to the process. Innovations that exploit the best software-engineering practices are identified and transferred throughout an organization.

Project teams in level 5 organizations analyze defects to determine their causes, evaluate the process to prevent known types of defects from recurring, and disseminate lessons learned to other projects.

Every system has chronic waste, in the form of rework, due to random variation in the tasks to be performed. At level 5, waste is unacceptable; organized efforts to remove waste result in changing the system by changing the *common* causes of inefficiency. Reducing waste happens at all the maturity levels, but it is the focus of level 5.

The software-process capability of level 5 organizations can be summarized as continuously improving because level 5 organizations are continuously striving to improve the range of their process capability, thereby improving the process performance of their projects. Improvement occurs both by incremental advancements in the existing process and by innovations in technologies and methods. Technology and process improvements are planned and managed as ordinary business activities.

Predicting performance. An organization's process maturity helps predict a project's ability to meet its goals. Projects in level 1 organizations experience wide variations in achieving cost, schedule, functionality, and quality targets.

As Figure 2 illustrates, we can expect three improvements in meeting targeted goals as an organization's process matures. These expectations are based on the quantitative results process improvement has achieved in other industries, and they are consistent with the initial case study results reported.[7-10]

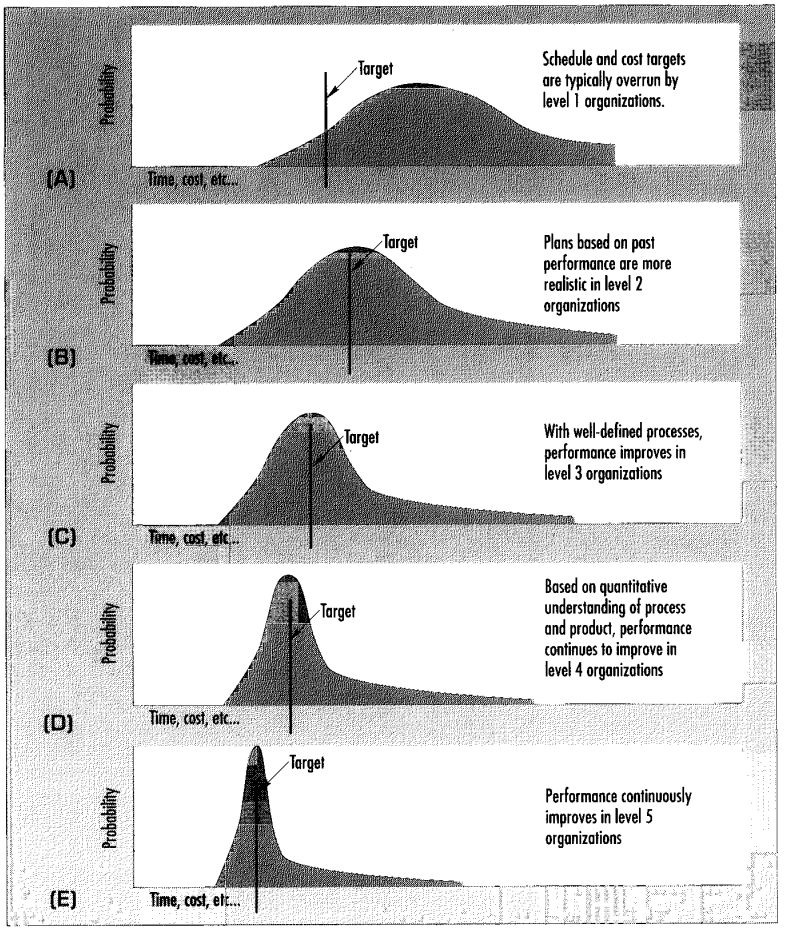


Figure 2. *Expected improvements in meeting goals (time, cost, etc.) as a process matures.* ***(A)*** *The curve in level 1 organizations is to the right of the target line because schedules and budgets are largely not met, and it covers a broad area because performance cannot be predicted;* ***(B)*** *more mature organizations can deliver projects of similar size and application in a smaller range and closer to the target;* ***(C)*** *still more mature organizations not only deliver more projects on target, but displace the target line horizontally, indicating shorter development time or reduced cost;* ***(D)*** *performance continues to improve because the process is adjusted based on quanitative data; and* ***(E)*** *reduced rework and improved predictability are key to further improvement.*

First, as maturity increases, the difference between targeted results and actual results decreases across projects. For example, level 1 organizations often miss their originally scheduled delivery dates by a wide margin. In Figure 2a, this is illustrated by how much of the area under the curve lies to the right of the target line. More mature organizations should be able to meet targeted dates with increased accuracy.

Second, as maturity increases, the variability of actual results around targeted results decreases. For example, in level 1 organizations, delivery dates for projects of similar size are unpredictable and vary widely. Similar projects in a more mature organization, however, will be delivered within a smaller range. In Figure 2b, this is illustrated by how much of the area under the curve is concentrated near the target line.

Third, as an organization matures, costs decrease, development time shortens, and productivity and quality increase. In a level 1 organization, development time can be quite long because of rework. In contrast, more mature organizations have increased process efficiency and reduced rework, shortening development time. In Figure 2c, this is illustrated by the horizontal displacement of the target line from the origin.

Improved prediction rests on the assumption that reducing noise, often in the form of rework, improves predictability. Unprecedented systems complicate the picture, because new technologies and applications lower process capability by in-

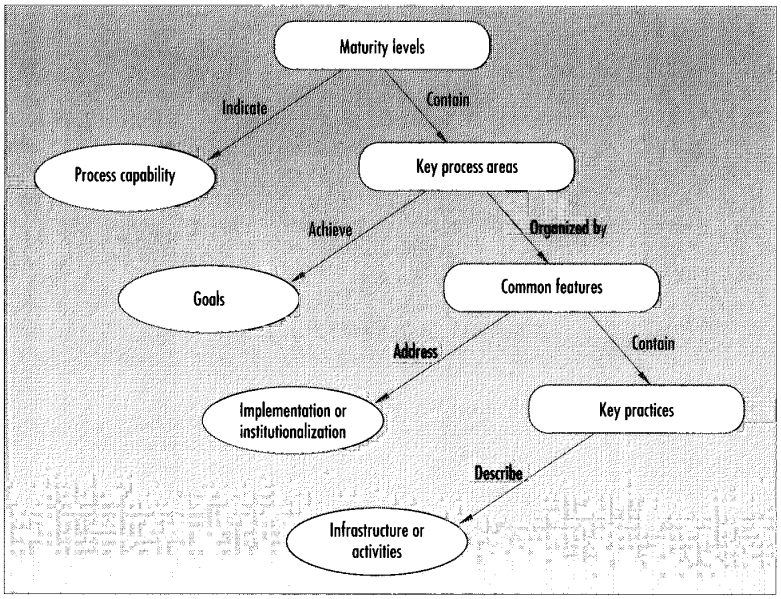


Figure 3. Each maturity level is composed of key process areas, which are composed of common features, which in turn specify key practices.

creasing variability.

Nevertheless, the management and engineering practices characteristic of more mature organizations help identify and address problems in unprecedented systems earlier in development than would be possible in less mature organizations. In some cases, a mature process means that "failed" projects are identified early, so investment in a lost cause is minimized.

The documented case studies of software-process improvement indicate that they result in significant improvements in both quality and productivity. The return on investment seems to be in the 5:1 to 8:1 range for successful process-improvement efforts.

Skipping levels. Trying to skip maturity levels is counterproductive because each level is a necessary foundation from which to achieve the next level. Organizations can institute specific process improvements at any time, even before they are prepared to advance to the level at which the practice is recommended. However, developers should understand that the stability of these improvements is at greater risk because they do not rest on a complete foundation. Processes without a proper foundation fail at the very time they are needed most — under stress — and provide no basis for future improvement.

For example, a well-defined level 3 process can be placed at great risk if management makes a poorly planned schedule commitment or fails to control changes to the baselined requirements. Similarly, many developers have collected the detailed data that is characteristic of level 4, only to find they cannot interpret it because their process is inconsistent.

At the same time, process improvement should focus on the needs of an organization in the context of its business environment. Higher level practices may address a project's or an organization's immediate needs. For example, one of the recommended steps to move from level 1 to level 2 is to establish a software-engineering process group, which is an attribute of level 3 organizations. So, while such a group is not a necessary characteristic of a level 2 organization, it can be useful in achieving level 2.

CMM OPERATIONAL DEFINITION

The CMM framework represents a path of improvements to increased software-process capability. The operational elaboration of the CMM is designed to support the many ways it will be used, four of which are

- Assessment teams will use it to identify strengths and weaknesses in an organization.
- Evaluation teams will use it to identify the risks of selecting among contractors and to monitor contracts.
- Upper management will use it to understand the activities necessary to launch a process-improvement program in their organization.
- Technical staff and process-improvement groups will use it as a guide to help them define and improve their organization's process.

Because these uses are diverse, the CMM must be decomposed in sufficient detail so that actual recommendations can be derived from it. This decomposition indicates the key processes and their structure that characterize software-process maturity and software-process capability.

Internal structure. The CMM decomposes each maturity level into constituent parts, with the exception of level 1. As Figure 3 shows, each level is composed of several *key process areas*. Each key process area is organized into five sections called *common features*. The common features specify *key practices*, which, when collectively addressed, accomplish the goals of the key process area.

Key process areas. Key process areas indicate where an organization should focus to improve its software process. They identify the issues that must be addressed to achieve a maturity level, as Figure 4 illustrates.

Each key process area identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing process capability. The path to achieving these goals may differ across projects, depending on the application domain or environment. Nevertheless, all the goals of a key process area must be achieved for an organization to satisfy it.

The use of the adjective "key" implies that there are process areas (and processes) that are not key to achieving a maturity level. The CMM does not describe in detail all the process areas involved with developing and maintaining software, only those that have been identified as key determiners of process capability.

Key process areas may be viewed as requirements for achieving a maturity level: To achieve a maturity level, the key process areas for that level must be satis-

fied. (Level 1 has no key process areas.)

The specific practices to be executed in each key process area will evolve as an organization achieves higher levels. For example, many of the project-estimating capabilities described in the project planning key process area at level 2 must evolve to handle the additional project data available at level 3.

Level 2. The key process areas at level 2 focus on establishing basic project-management controls.

♦ *Requirements Management* means establishing a common understanding between a customer and a project team of the customer's requirements. This agreement is the basis for planning and managing a project.

♦ *Software Project Planning* means establishing reasonable plans for engineering and managing a project. These plans are the foundation project management.

♦ *Software Project Tracking and Oversight* means to establish adequate visibility into actual progress so that management can take effective action when a project's performance deviates significantly from the plans.

♦ *Software Subcontract Management* means to select qualified subcontractors and manage them effectively.

♦ *Software Quality Assurance* means to provide management with appropriate visibility into the process being used and the products being built.

♦ *Software Configuration Management* means to establish and maintain the integrity of a project's products throughout its life cycle.

Level 3. The key process areas at level 3 address both project and organizational issues, as an organization establishes an infrastructure that institutionalizes effective software-engineering and management processes across all projects.

♦ *Organization Process Focus* means to establish an organizational responsibility for activities that improve an organization's overall software-process capability.

♦ *Organization Process Definition* means to develop and maintain a usable set of process assets that improve processes across projects and provide a basis for defining meaningful data for quantitative process management. These assets are a foundation that can be institutionalized through mechanisms like training.

♦ *Training Program* means to develop the skills and knowledge of individuals so that they can be effective and efficient. Training is an organizational responsibility, but projects should identify necessary skills and provide training when their needs are unique.

♦ *Integrated Software Management* means to integrate software-engineering and management activities into a coherent, defined process that is tailored from an organization's standard software process and related process assets. Tailoring is based on the business environment and technical needs of a project.

♦ *Software Product Engineering* means to consistently perform a well-defined process that integrates all technical activities — requirements analysis, design, code, and test, among others — to produce correct, consistent software products effectively and efficiently.

♦ *Intergroup Coordination* means to establish a way for a software-engineering group to participate actively with other engineering groups so that a project team can better satisfy the customer's needs.

♦ *Peer Reviews* means to remove defects from work products early and efficiently. An important corollary effect is to develop a better understanding of the work products and preventable defects. Peer review is an important and effective method that can be implemented through inspections or structured walkthroughs, for example.

Level 4. The key process areas at level 4 focus on establishing a quantitative understanding of both the software process and the software work products being built.

♦ *Quantitative Process Management* means to control a project's process performance quantitatively. A project's pro-

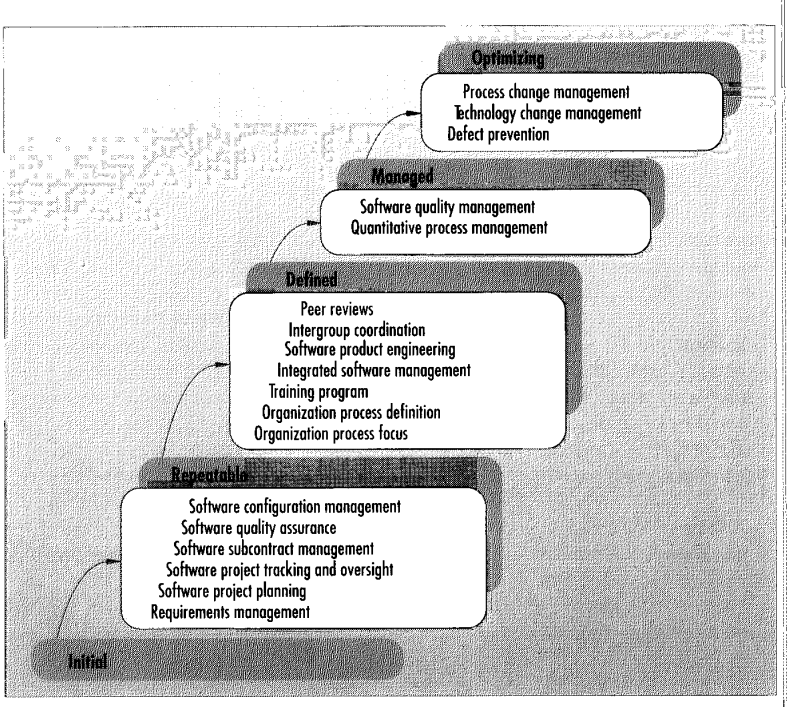
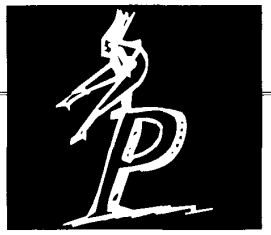


Figure 4. *Key process areas identify goals that must be met to achieve a maturity level. The path to achieving these goals may differ across projects.*



cess performance comprises the actual results achieved from following a software process. The focus is on identifying special causes of variation within a measurably stable process and correcting, as appropriate, the circumstances that created them.

♦ *Software Quality Management* means to develop a quantitative understanding of the quality of a project's products to achieve specific quality goals.

Level 5. The key process areas at level 5 focus on issues that both organizations and projects must address to implement continuous and measurable process improvement.

♦ *Defect Prevention* means to identify the causes of defects and prevent them from recurring by analyzing them and changing the defined process.

♦ *Technology Change Management* means to identify beneficial new technologies (such as tools, methods, and processes) and transfer them into an organization in an orderly manner. The focus here is on efficient innovation in an ever-changing world.

♦ *Process Change Management* means to continually improve an organization's processes with the intent of improving quality, increasing productivity, and decreasing development time.

Goals. Goals, which summarize key practices, are used to determine if an organization or project has effectively implemented a key process area. The goals signify the scope, boundaries, and intent of each key process area.

Common features. For convenience, the practices that describe the key process areas are organized by common features. The common features are attributes that indicate whether the implementation and institutionalization of a key process area is effective, repeatable, and lasting.

The five common features are

♦ *Commitment to perform*, which describes the actions an organization must take to ensure its process is established and enduring. This typically involves establishing organizational policies and obtaining senior-management sponsorship.

KEY PRACTICES DESCRIBE WHAT TO DO, BUT THEY DO NOT MANDATE HOW TO DO IT.

♦ *Ability to perform*, which describes the preconditions in a project or organization to implement the process competently. This typically involves resources, organizational structures, and training.

♦ *Activities performed*, which describes the roles and procedures necessary to implement a key process area. This typically involves establishing plans and procedures, performing the work, tracking it, and taking corrective actions as necessary.

♦ *Measurement and analysis*, which describes the need to measure the process and analyze the measurements. This typically involves obtaining sample measurements that could determine the status and effectiveness of activities performed.

♦ *Verifying implementation*, which describes the steps to ensure that the activities are performed in compliance with the standard process. This typically involves reviews and audits by management and quality assurance.

Activities performed describes what must be implemented to establish a process capability; the others, taken as a whole, are the basis by which an organization can institutionalize activities performed.

Key practices. Each key process area is described in terms of key practices that contribute to satisfying its goals. Key practices describe the infrastructure and activities that contribute most to the effective implementation and institutionalization of the key process area.

Each key practice consists of a single sentence, often followed by a more detailed description, which may include examples and elaboration. These key practices, also called top-level key practices, state the fundamental policies, procedures, and activities for the key process area. The components of the detailed description are frequently referred to as subpractices.

The key practices describe *what* to do, but they do not mandate *how* to do it. Alternative practices may also accomplish the goals of the key process area. The key practices should be interpreted rationally, to judge if the goals of the key process area are effectively, although perhaps differently, achieved. The key practices are contained in a separate report, along with guidance on their interpretation.[6]

FUTURE DIRECTIONS

Achieving higher levels of software-process maturity is incremental and requires a long-term commitment to continuous process improvement. Software organizations may take 10 years or more to build the foundation for, and a culture oriented toward, continuous process improvement. Although a decade-long process improvement program is foreign to most US companies, this level of effort is required to produce mature software organizations.

The CMM is not a silver bullet and does not address all the issues important for successful projects. For example, it does not currently address expertise in particular application domains, advocate specific software technologies, or suggest how to select, hire, motivate, and retain competent people -- although these issues are crucial to a project's success.

During the next few years, the CMM will continue to undergo extensive testing through use in software-process assessments, software-capability evaluations, and process-improvement programs. CMM-based products and training materials will be developed and revised as appropriate. The CMM is a living document that will be improved, but the SEI anticipates that CMM version 1.1 will remain the baseline until at least 1996. This provides an appropriate and realistic balance between the need for stability and the goal of continuous improvement. A book on the CMM is in progress for the SEI series published by Addison-Wesley.

The SEI is also working with the International Standards Organization in its efforts to build international standards for software-process assessment, improvement, and capability evaluation. This effort will integrate concepts from many process-improvement methods. The development of the ISO standards (and the contributions of other methods) will influence CMM version 2.0, even as the SEI's process work will influence the activities of the ISO.

The CMM represents a common-sense-engineering approach to software-process improvement. The maturity levels, key process areas, common features, and key practices have been extensively discussed and reviewed within the software community. 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.

The CMM provides a conceptual structure for improving the management and development of software products in a disciplined and consistent way. It does not guarantee that software products will be successfully built or that all problems in software engineering will be adequately resolved. However, current reports from CMM-based improvement programs indicate that it can improve the likelihood with which a software organization can achieve its cost, quality, and productivity goals.[7-10]

The CMM identifies practices for a mature software process and provides examples of the state of the practice (and in some cases, the state of the art), but it is not meant to be either exhaustive or dictatorial. The CMM identifies the characteristics of an effective software process, but the mature organization addresses all issues essential to a successful project, including people and technology, as well as process. ♦

Mark C. Paulk, a member of the technical staff at the Software Engineering Institute, is the project leader for the Capability Maturity Model project, which develops products for software-process determination and improvement.

Paulk received a BS in mathematics from the University of Alabama, Huntsville, and an MS in computer science from Vanderbilt University. He is a senior member of the IEEE and a member of the American Society for Quality Control.

Mary Beth Chrissis, a member of the SEI technical staff, has been involved in the initial development and revision of the CMM. She is interested in developing methods and tools that will help organizations improve their software processes.

Chrissis received a BS in technical writing from Carnegie Mellon University and has pursued postgraudate studies in computer science at Johns Hopkins University.

Bill Curtis is the former director of the software-process program at the SEI, where he continues to work on developing a human-resource maturity model. He is also a founding faculty member of the Software Quality Institute at the University of Texas at Austin, and works with organizations to increase their software-development capability.

Curtis serves on the editorial boards of several technical journals and edits *IEEE Software's* Interface department. He is a member of the IEEE, ACM, American Psychological Association, Human Factors Association, and American Association of Artificial Intelligence.

Charles V. Weber is a member of the system-development-process group at IBM Federal Systems Co., Boulder, Colorado. As an SEI resident affiliate, he was the primary author of the practices of CMM version 1.0; he contributed significantly to version 1.1.

Weber received a BS in mathematics from the University of Minnesota in Minneapolis. He chairs the SEI's CMM Advisory Board.

For information about the CMM, training, and performing process assessments and capability evaluations, contact SEI Customer Relations, Software Eng. Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890; Internet: customer-relations@sei.cmu.edu.

ACKNOWLEDGMENTS

The CMM was produced by a dedicated group of people who spent many hours discussing the model and its features and then documenting it. Contributors, other than ourselves, include Edward Averill, Judy Bamberger, Joe Besselman, Marilyn Bush, Anita Carleton, Marty Carlson, Susan Dart, Betty Deimel, Lionel Deimel, Peter Feiler, Julia Gale, Suzie Garcia, Jim Hart, Ron Higuera, Watts Humphrey, Purvis Jackson, Tim Kasse, Richard Kauffold, David Kitson, Mike Konrad, Peter Malpass, Mark Manduke, Steve Masters, Mary Merrill, Judah Mogilensky, Warren Moseley, Jim Over, George Pandelios, Bob Park, Jeff Perdue, Dick Phillips, Mike Rissman, Jim Rozum, Jane Siegel, Christer von Schantz, Cynthia Wise, and Jim Withey.

We appreciate the administrative help from Todd Bowman, Dorothy Josephson, Debbie Punjack, Carolyn Tady, Marcia Theoret, Andy Tsounos, and David White and the editorial assistance from Suzanne Couturiaux and Bill Pollak.

Special thanks go to the members of the CMM Correspondence Group, who contributed their time and effort to reviewing drafts of the CMM and providing insightful comments and recommendations, and to the members of the CMM Advisory Board, who helped guide us. The current members of the Advisory Board are Constance Ahara, Kelley Butler, Bill Curtis, Conrad Czaplicki, Raymond Dion, Judah Mogilensky, Martin Owens, Mark Paulk, Sue Stetak, Charles Weber, and Ron Willis. Former members who worked with us on CMM 1.0 include Harry Carl, Jim Hess, Jerry Pixton, and Jim Withey.

This work was sponsored by the US Department of Defense.

REFERENCES

1. W.S. Humphrey, "Characterizing the Software Process: A Maturity Framework," Tech. Report CMU/SEI-87-TR-11, ADA182895, Software Eng. Institute, Pittsburgh, June 1987; also in *IEEE Software*, Mar. 1988, pp. 73-79.
2. W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, Reading, Mass., 1989.
3. W.S. Humphrey and W.L. Sweet, "A Method for Assessing the Software Engineering Capability of Contractors," Tech. Report CMU/SEI-87-TR-23, ADA187320, Software Eng. Institute, Pittsburgh, 1987.
4. M.C. Paulk et al., "Capability Maturity Model for Software, Version 1.1," Tech. ReportCMU/SEI-93-TR-24, Software Eng. Institute, 1993.
5. P. Fowler and S. Rifkin, "Software Engineering Process Group Guide," Tech. Report CMU/SEI-90-TR-24, ADA235784, Software Eng. Institute, Pittsburgh, 1990.
6. M.C. Paulk et al., "Key Practices of the Capability Maturity Model, Version 1.1," Tech. Report CMU/SEI-93-TR-25, Software Eng. Institute, Pittsburgh, 1993.
7. R. Dion, "Elements of a Process-Improvement Program," *IEEE Software*, July 1992, pp. 83-85.
8. W.S. Humphrey, T.R. Snyder, and R.R. Willis, "Software Process Improvement at Hughes Aircraft," *IEEE Software*, July 1991, pp. 11-23.
9. W.H. Lipke and K.L. Butler, "Software Process Improvement: A Success Story," *Crosstalk*, Nov. 1992, pp. 29-31.
10. D.H. Kitson and S. Masters, "An Analysis of SEI Software Process Assessment Results: 1987-1991," Tech. Report CMU/SEI-92-TR-24, Software Eng. Institute, Pittsburgh, 1992.