The Capability Maturity Model for Software

Mark C. Paulk
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Bill Curtis
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Mary Beth Chrissis
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Charles V. Weber
IBM Federal Systems Company
6300 Diagonal Highway
Boulder, CO 80301

Abstract

This paper provides an overview of the latest version of the Capability Maturity Model for Software, CMM v1.1. Based on over six years of experience with software process improvement and the contributions of hundreds of reviewers, CMM v1.1 describes the software engineering and management practices that characterize organizations as they mature their processes for developing and maintaining software. This paper stresses the need for a process maturity framework to prioritize improvement actions, describes the process maturity framework of five maturity levels and the associated structural components, and discusses future directions for the CMM.

Keywords: capability maturity model, CMM, process maturity framework, software process improvement, process capability, process performance, maturity level, key process area, software process assessment, software capability evaluation.

1 Introduction

After two decades of unfulfilled promises about productivity and quality gains from applying new software methodologies and technologies, organizations are realizing that their fundamental problem is the 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 (SEI), with assistance from the Mitre Corporation, began developing a process maturity framework that would help organizations improve their software process. In September 1987, the SEI released a brief description of the process maturity framework [Humphrey 87a] which was later expanded in Humphrey's book, *Managing the Software Process* [Humphrey89]. Two methods, software process assessment1 and software capability evaluation2 and a maturity questionnaire [Humphrey87b] were developed to appraise software process maturity.

After four years of experience with the software process maturity framework and the preliminary version of the maturity questionnaire, the SEI evolved the maturity framework into the Capability Maturity Model for Software (CMM) [Paulk91, Weber91]. The CMM presents sets of recommended practices in a number of key process areas that have been shown to enhance software process capability. The CMM is based on knowledge acquired from software process assessments and extensive feedback from both industry and government.

The Capability Maturity Model for Software provides software organizations with guidance on how to gain control of their processes for developing and maintaining software and how to evolve toward a culture of software engineering and management excellence. The CMM was designed to guide software organizations in selecting process improvement strategies by determining current process maturity and identifying the few issues most

1 A software process assessment is an appraisal by a trained team of software professionals to determine the state of an organization's current software process, to determine the high-priority software process-related issues facing an organization, and to obtain the organizational support for software process improvement.

2 A software capability evaluation is an appraisal by a trained team of professionals to identify contractors who are qualified to perform the software work or to monitor the state of the software process used on an existing software effort.

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

The initial release of the CMM, v1.0, was reviewed and used by the software community during 1991 and 1992. A workshop was held in April, 1992 on CMM v1.0, and was attended by about 200 software professionals. The current version of the CMM, v1.1 [Paulk93a, Paulk93b], is the result of the feedback from that workshop and ongoing feedback from the software community.

1.1 Immature Versus Mature Software Organizations

Setting sensible goals for process improvement requires an understanding of the difference between immature and mature software organizations. In an immature software organization, software processes are generally improvised by practitioners and their management during the course of the project. Even if a software process has been specified, it is not rigorously followed or enforced. The immature software organization is reactionary, and 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 the schedule.

In an immature organization, there is no objective basis for judging product quality or for solving 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.

On the other hand, a mature software organization possesses an organization-wide ability for managing software development and maintenance processes. The software process is accurately communicated to both existing staff and new employees, and work activities are carried out according to the planned process. The processes mandated are usable and consistent with the way the work actually gets done. These defined processes are updated when necessary, and improvements are developed through controlled pilot-tests and/or cost benefit analyses. Roles and responsibilities within the defined process are clear throughout the project and across the organization.

In a mature organization, managers monitor the quality of the software 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; the expected results for cost, schedule, functionality, and quality of the product are usually achieved. In general, a disciplined process is consistently followed because all of the participants understand the value of doing so, and the necessary infrastructure exists to support the process.

1.2 Fundamental Concepts Underlying Process Maturity

A ***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 (e.g., project plans, design documents, code, test cases, and user manuals). 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. The software process capability of an organization provides one means of predicting the most likely outcomes to be expected from the next software project the organization undertakes.

Software process performance represents the actual results achieved by following a software process. Thus, software process performance focuses on the results achieved, while software process capability focuses on results expected.

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 software process 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.

2 The Five Levels of Software Process Maturity

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 that lay successive foundations for continuous process improvement. These five maturity levels define an ordinal scale for measuring the maturity of an organization's software process and for evaluating its software process capability. The levels also help an organization prioritize its improvement efforts.

A ***maturity level*** is a well-defined evolutionary plateau toward achieving a mature software process. Each maturity level comprises a set of process goals that, when satisfied, stabilize an important component of the software process. Achieving each level of the maturity framework establishes a different component in the software process, resulting in an increase in the process capability of the organization.

Organizing the CMM into the five levels shown in Figure 2.1 prioritizes improvement actions for increasing software process maturity. The labeled arrows in Figure 2.1 indicate the type of process capability being institutionalized by the organization at each step of the maturity framework.

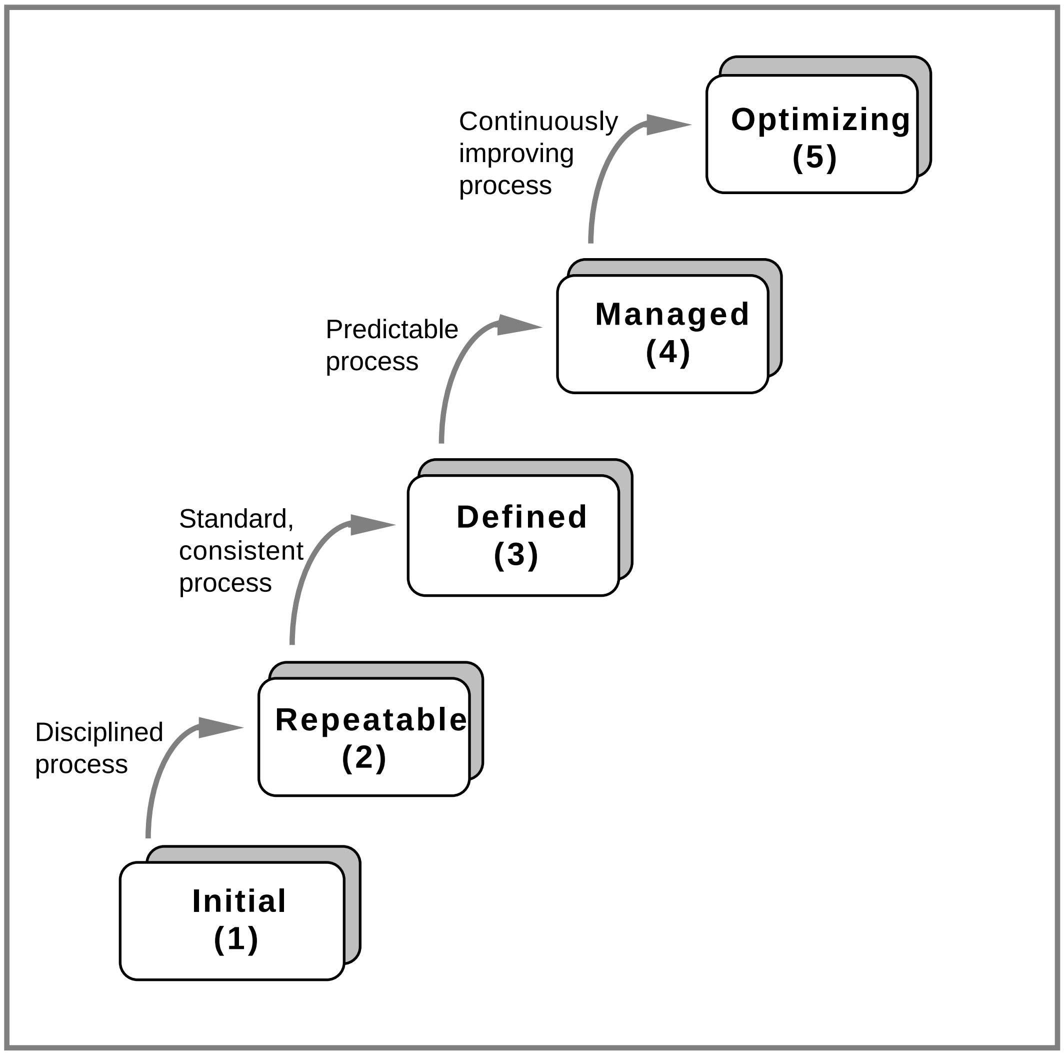


Figure 2.1 The Five Levels of Software Process Maturity

2.1 Behavioral Characterization of the Maturity Levels

Maturity Levels 2 through 5 can be characterized through the activities performed by the organization to establish or improve the software process, by activities performed on each project, and by the resulting process capability across projects. A behavioral characterization of Level 1 is included to establish a base of comparison for process improvements at higher maturity levels.

2.1.1 Level 1 - The Initial Level

At the Initial Level, the 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 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 software team. Occasionally, capable and forceful software managers can withstand the pressures to take shortcuts in the software process; 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.

In spite of this ad hoc, even chaotic, process, Level 1 organizations frequently develop products that work, even though they may be over the budget and schedule. Success in Level 1 organizations depends on the competence and heroics of the people in the organization[3] and cannot be repeated unless the same competent individuals are assigned to the next project. Thus, at Level 1, capability is a characteristic of the individuals, not of the organization.

2.1.2 Level 2 - The Repeatable Level

At the Repeatable Level, policies for managing a software project and procedures to implement those policies are established. Planning and managing new projects is based on experience with similar projects. Process capability is enhanced by establishing basic process management discipline on a project by project basis. An effective process can be characterized as one which is practiced, documented, enforced, trained, measured, and able to improve.

Projects in Level 2 organizations have installed basic software management controls. Realistic project commitments are based on the results observed on previous projects and on the requirements of the current project. The software managers for a project track software costs, schedules, and functionality; problems in meeting commitments are identified when they arise. Software requirements and the work products developed to satisfy them are baselined,

[3] Selecting, hiring, developing, and/or retaining competent people are significant issues for organizations at all levels of maturity, but they are largely outside the scope of the CMM.

and their integrity is controlled. Software project standards are defined, and the organization ensures they are faithfully followed. The software project works with its subcontractors, if any, to establish a customer-supplier relationship.

Processes may differ between projects in a Level 2 organization. The organizational requirement for achieving Level 2 is that there are policies that guide the projects in establishing the appropriate management processes.

The software process capability of Level 2 organizations can be summarized as disciplined because planning and tracking of the software project is stable and earlier successes can be repeated. The project's process is under the effective control of a project management system, following realistic plans based on the performance of previous projects.

2.1.3 Level 3 - The Defined Level

At the Defined Level, the standard 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. This standard process is referred to throughout the CMM as the organization's standard software process. Processes established at Level 3 are used (and changed, as appropriate) to help the software managers and technical staff perform more effectively. The organization exploits effective software engineering practices when standardizing its software processes. There is a group that is responsible for the organization's software process activities, e.g., a software engineering process group, or SEPG [Fowler90]. An organization-wide training program is implemented to ensure that the staff and managers have the knowledge and skills required to fulfill their assigned roles.

Projects tailor the organization's standard software process to develop their own defined software process, which accounts for the unique characteristics of the project. This tailored process is referred to in the CMM as the project's defined software process. A defined software process contains a coherent, integrated set of well-defined software engineering and management processes.   A well-defined process can be characterized as including readiness criteria, inputs, standards and procedures for performing the work, verification mechanisms (such as peer reviews), outputs, and completion criteria. Because the software 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 established product lines, cost, schedule, and functionality are under control, and software quality is tracked. This process capability is based on a common, organization-wide understanding of the activities, roles, and responsibilities in a defined software process.

2.1.4 Level 4 - The Managed Level

At the Managed Level, the organization sets quantitative quality goals for both software products and processes. Productivity and quality are measured for important software process activities across all projects as part of an organizational measurement program. An organization-wide software process database is used to collect and analyze the data available from the projects' defined software processes. Software processes are instrumented with well-defined and consistent measurements at Level 4. These measurements establish the quantitative foundation for evaluating the projects' software processes and products.

Projects achieve control over their products and processes by narrowing the variation in their process 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 process capability allows an organization to 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, the "special cause" of the variation can be identified and addressed. When the known limits of the process are exceeded, action is taken to correct the situation. Software products are of predictably high quality.

2.1.5 Level 5 - The Optimizing Level

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 the occurrence of defects. Data on the effectiveness of the software process is used to perform cost benefit analyses of new technologies and proposed changes to the organization's software process. Innovations that exploit the best software engineering practices are identified and transferred throughout the organization.

Software project teams in Level 5 organizations analyze defects to determine their causes. Software processes are evaluated to prevent known types of defects from recurring, and lessons learned are disseminated to other projects.

There is chronic waste, in the form of rework, in any system simply due to random variation. Waste is unacceptable; organized efforts to remove waste result in changing the system, i.e., improving the process by changing "common causes" of inefficiency to prevent the waste from occurring. While this is true of all the maturity levels, it is the focus of Level 5.

The software process capability of Level 5 organizations can be characterized 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 using new technologies and methods. Technology and process improvements are planned and managed as ordinary business activities.

2.2 Process Capability and the Prediction of Performance

The maturity of an organization's software process helps to 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 illustrated in Figure 2.4, three improvements in meeting targeted goals are expected as the organization's software 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 from software organizations [Dion92, Humphrey91b, Lipke92, Wohlwend93].

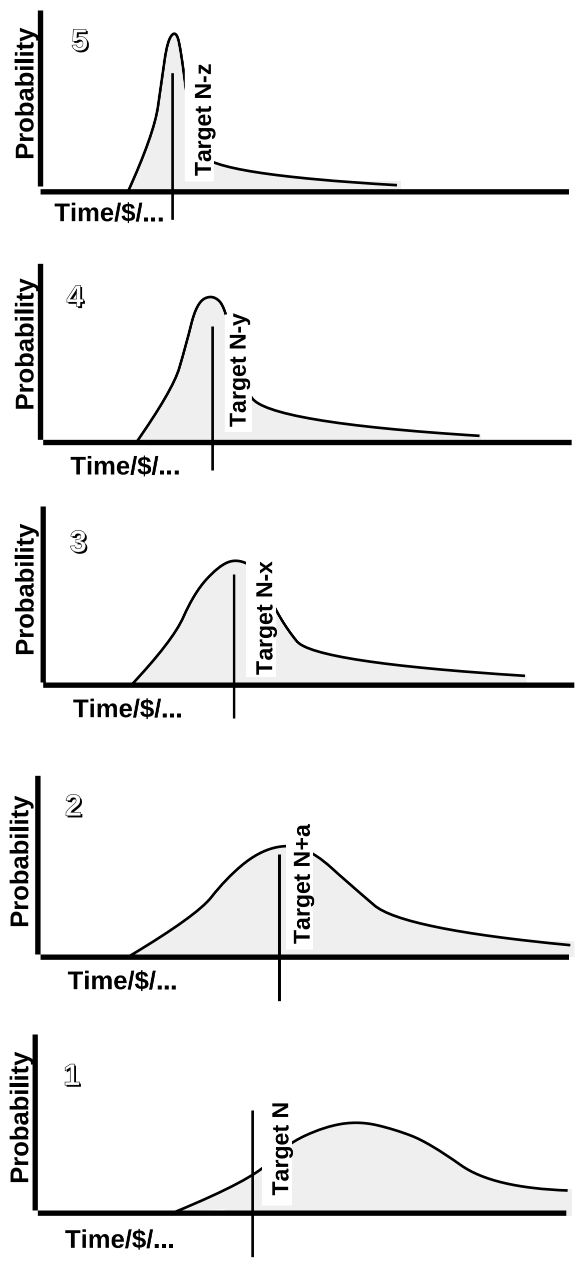
First, as maturity increases, the difference between targeted results and actual results decreases across projects. For instance, Level 1 organizations often miss their originally scheduled delivery dates by a wide margin, whereas higher maturity level organizations should be able to meet targeted dates with increased accuracy. (This is illustrated in Figure 2.4 by how much of the area under the curve lies to the right of the target line.)

Second, as maturity increases, the variability of actual results around targeted results decreases. For instance, in Level 1 organizations delivery dates for projects of similar size are unpredictable and vary widely. Similar projects in a higher maturity level organization, however, will be delivered within a smaller range. (This is illustrated in Figure 2.4 by how much of the area under the curve is concentrated near the target line.)

Third, targeted results improve as the maturity of the organization increases. That is, as a software organization matures, costs decrease, development time becomes shorter, and productivity and quality increase. In a Level 1 organization, development time can be quite long because of the amount of rework that must be performed to correct mistakes. In contrast, higher maturity level organizations have increased process efficiency and reduce costly rework, allowing development time to be shortened. (This is illustrated in Figure 2.4 by the horizontal displacement of the target line from the origin.)

The improvements in predicting a project's results represented in Figure 2.4 assume that the software project's outcomes become more predictable as noise, often in the form of rework, is removed from the software process. Unprecedented systems complicate the picture since new technologies and applications lower the process capability by increasing variability. Even in the case of unprecedented systems, the management and engineering practices characteristic of more mature organizations help identify and address problems earlier in the development cycle than they would have been detected in less mature organizations. In some cases a mature process means that "failed" projects are identified early in the software life cycle and investment in a lost cause is minimized.

The documented case studies of software process improvement indicate that there are significant improvements in both quality and productivity as a result of the improvement effort [Dion92, Humphrey91b, Lipke92, Wohlwend93]. The return on investment seems to typically be in the 5:1 to 8:1 range for successful process improvement efforts.



Performance continuously improves in Level 5 organizations

Based on quantitative understanding of process and product, performance continues to improve in Level 4 organizations

With well-defined processes, performance improves in Level 3 organizations

Plans based on past performance are more realistic in Level 2 organizations

Schedule and cost targets are typically overrun by Level 1 organizations.

Figure 2.4 Process Capability as Indicated by Maturity Level

2.3 Skipping Maturity Levels

Trying to skip levels is counterproductive because each maturity level in the CMM forms a necessary foundation from which to achieve the next level. The CMM identifies the levels through which an organization should evolve to establish a culture of software engineering excellence. Organizations can institute specific process improvements at any time they choose, even before they are prepared to advance to the level at which the specific practice is recommended. However, organizations should understand that the stability of these improvements is at greater risk since the foundation for their successful institutionalization has not been completed. Processes without the proper foundation fail at the very point they are needed most – under stress – and they provide no basis for future improvement.

For instance, a well-defined software process that is characteristic of a Level 3 organization, 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 organizations have collected the detailed data characteristic of Level 4, only to find that the data were uninterpretable because of inconsistency in the software development processes.

At the same time, it must be recognized that process improvement efforts should focus on the needs of the organization in the context of its business environment, and higher-level practices may address the current needs of an organization or project. For example, when prescribing what steps an organization should take to move from Level 1 to Level 2, frequently one of the recommendations is to establish a software engineering process group (SEPG), which is an attribute of Level 3 organizations. While an SEPG is not a necessary characteristic of a Level 2 organization, they can be a useful part of the prescription for achieving Level 2.

3 Operational Definition of the Capability Maturity Model

The CMM is a framework representing a path of improvements recommended for software organizations that want to increase their software process capability. This operational elaboration of the CMM is designed to support the many ways it will be used. There are at least four uses of the CMM that are supported:

- Assessment teams will use the CMM to identify strengths and weaknesses in the organization.
- Evaluation teams will use the CMM to identify the risks of selecting among different contractors for awarding business and to monitor contracts.
- Upper management will use the CMM to understand the activities necessary to launch a software process improvement program in their organization.
- Technical staff and process improvement groups, such as an SEPG, will use the CMM as a guide to help them define and improve the software process in their organization.

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

3.1 Internal Structure of the Maturity Levels

Each maturity level has been decomposed into constituent parts. With the exception of Level 1, the decomposition of each maturity level ranges from abstract summaries of each level down to their operational definition in the key practices, as shown in Figure 3.1. Each maturity level is composed of several key process areas. Each key process area is organized into five sections called common features. The common features specify the key practices that, when collectively addressed, accomplish the goals of the key process area.

3.2 Maturity Levels

A maturity level is a well-defined evolutionary plateau toward achieving a mature software process. Each maturity level indicates a level of process capability, as was illustrated in Figure 2.1. For instance, at Level 2 the process capability of an organization has been elevated from ad hoc to disciplined by establishing sound project management controls.

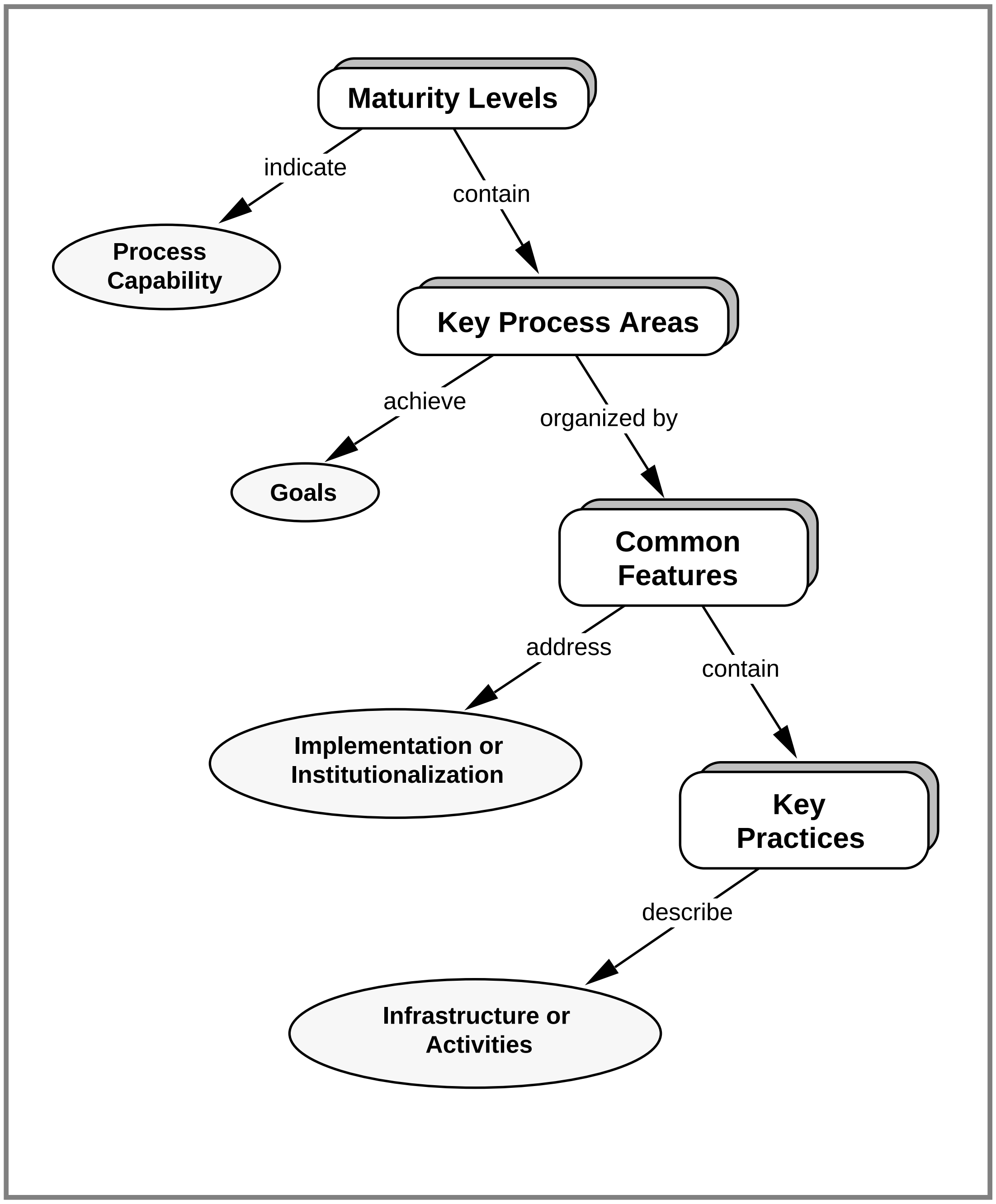


Figure 3.1 The CMM Structure

3.3 Key Process Areas

Except for Level 1, each maturity level is decomposed into several key process areas that indicate where 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.

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 key process areas have been defined to reside at a single maturity level as shown in Figure 3.2. The path to achieving the goals of a key process area may differ across projects based on differences in application domains or environments. Nevertheless, all the goals of a key process area must be achieved for the organization to satisfy that key process area.

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 all the process areas in detail that are involved with developing and maintaining software. Certain process areas have been identified as key determiners of process capability; these are the ones described in the CMM.

The key process areas may be considered the requirements for achieving a maturity level. To achieve a maturity level, the key process areas for that level must be satisfied.

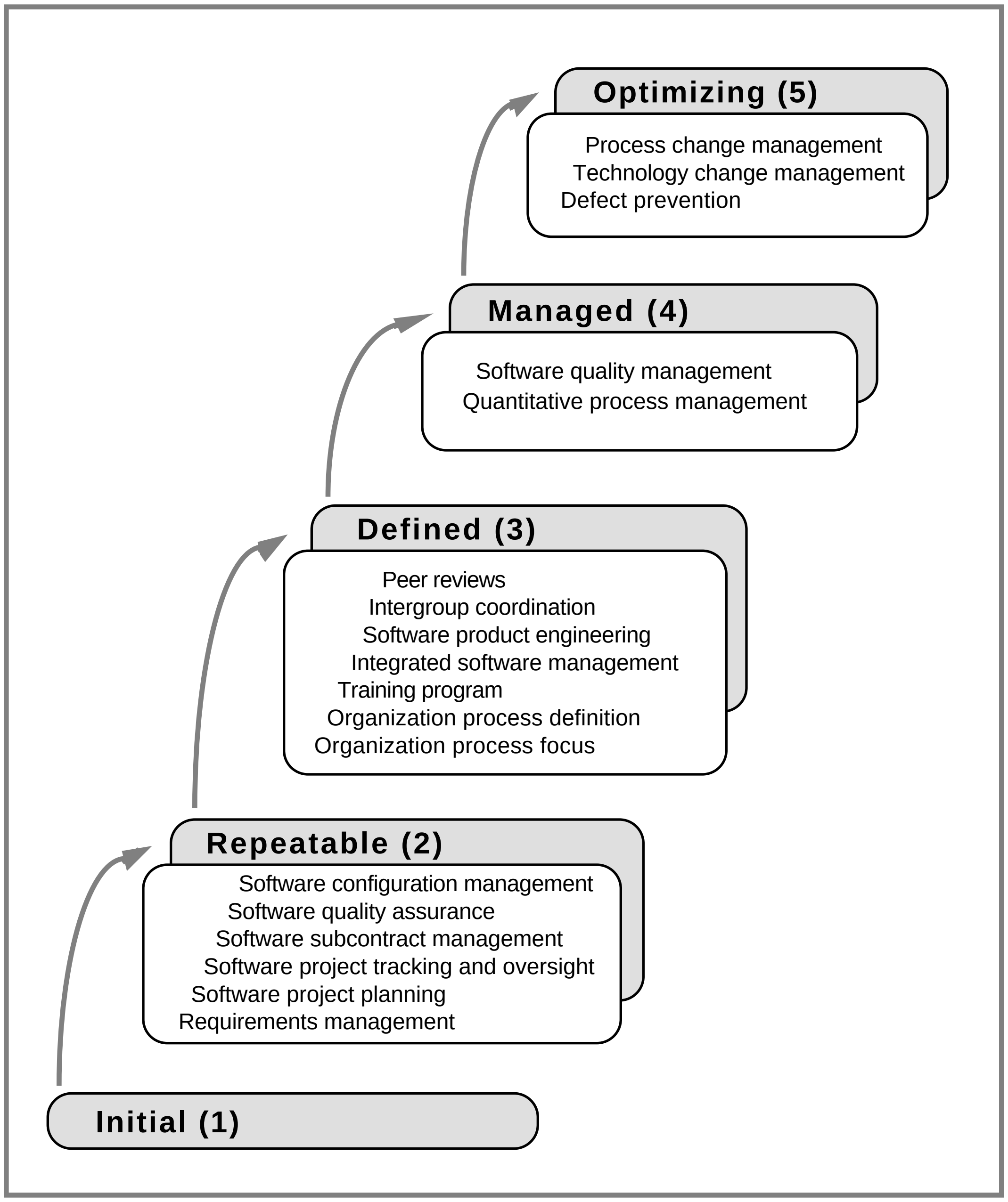

Optimizing (5)
Process change management
Technology change management
Defect prevention
Managed (4)
Software quality management
Quantitative process management
Defined (3)
Peer reviews
Intergroup coordination
Software product engineering
Integrated software management
Training program
Organization process definition
Organization process focus
Repeatable (2)
Software configuration management
Software quality assurance
Software subcontract management
Software project tracking and oversight
Software project planning
Requirements management
Initial (1)

Figure 3.2 The Key Process Areas by Maturity Level

The specific practices to be executed in each key process area will evolve as the organization achieves higher levels of process maturity. For instance, many of the project estimating capabilities described in the Software Project Planning key process area at Level 2 must evolve to handle the additional project data available at Level 3, as is described in Integrated Software Management.

The key process areas at Level 2 focus on the software project's concerns related to establishing basic project management controls.

° The purpose of Requirements Management is to establish a common understanding between the customer and the software project of the customer's requirements that will be addressed by the software project. This agreement with the customer is the basis for planning and managing the software project.

° The purpose of Software Project Planning is to establish reasonable plans for performing the software engineering and for managing the software project. These plans are the necessary foundation for managing the software project.

° The purpose of Software Project Tracking and Oversight is to establish adequate visibility into actual progress so that management can take effective actions when the software project's performance deviates significantly from the software plans.

° The purpose of Software Subcontract Management is to select qualified software subcontractors and manage them effectively.

° The purpose of Software Quality Assurance is to provide management with appropriate visibility into the process being used by the software project and of the products being built.

° The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle.

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

- The purpose of Organization Process Focus is to establish the organizational responsibility for software process activities that improve the organization's overall software process capability.

- The purpose of Organization Process Definition is to develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for defining meaningful data for quantitative process management. These assets provide a stable foundation that can be institutionalized via mechanisms such as training.

- The purpose of Training Program is to develop the skills and knowledge of individuals so they can perform their roles effectively and efficiently. Training is an organizational responsibility, but the software projects should identify their needed skills and provide the necessary training when the project's needs are unique.

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

- The purpose of Software Product Engineering is to consistently perform a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently. Software Product Engineering describes the technical activities of the project, e.g., requirements analysis, design, code, and test.

- The purpose of Intergroup Coordination is to establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently.

- The purpose of Peer Reviews is to remove defects from the software work products early and efficiently. An important corollary effect is to develop a better understanding of the software work products and of the defects that can be prevented. The peer review is an important and effective engineering method that can be implemented via inspections, structured walkthroughs, or a number of other collegial review methods.

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.

- The purpose of Quantitative Process Management is to control the process performance of the software project quantitatively. Software process performance represents 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 drove the transient variation to occur.
- The purpose of Software Quality Management is to develop a quantitative understanding of the quality of the project's software products and achieve specific quality goals.

The key process areas at Level 5 cover the issues that both the organization and the projects must address to implement continuous and measurable software process improvement.

- The purpose of Defect Prevention is to identify the causes of defects and prevent them from recurring. The software project analyzes defects, identifies their causes, and changes its defined software process.
- The purpose of Technology Change Management is to identify beneficial new technologies (i.e., tools, methods, and processes) and transfer them into the organization in an orderly manner. The focus of Technology Change Management is on performing innovation efficiently in an ever-changing world.
- The purpose of Process Change Management is to continually improve the software processes used in the organization with the intent of improving software quality, increasing productivity, and decreasing the cycle time for product development.

3.4 Goals

The ***goals*** summarize the key practices of a key process area and can be used to determine whether an organization or project has effectively implemented the key process area. The goals signify the scope, boundaries, and intent of each key process area. Satisfaction of a KPA is determined by achievement of the goals.

3.5 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	Commitment to Perform describes the actions the organization must take to ensure that the process is established and will endure. Commitment to Perform typically involves establishing organizational policies and senior management sponsorship.
Ability to Perform	Ability to Perform describes the preconditions that must exist in the project or organization to implement the software process competently. Ability to Perform typically involves resources, organizational structures, and training.
Activities Performed	Activities Performed describes the roles and procedures necessary to implement a key process area. Activities Performed typically involve establishing plans and procedures, performing the work, tracking it, and taking corrective actions as necessary.
Measurement and Analysis	Measurement and Analysis describes the need to measure the process and analyze the measurements. Measurement and Analysis typically includes examples of the measurements that could be taken to determine the status and effectiveness of the Activities Performed.
Verifying Implementation	Verifying Implementation describes the steps to ensure that the activities are performed in compliance with the process that has been established. Verification typically encompasses reviews and audits by management and software quality assurance.

The practices in the common feature Activities Performed describe what must be implemented to establish a process capability. The other practices, taken as a whole, form the basis by which an organization can institutionalize the practices described in the Activities Performed common feature.

3.6 Key Practices

Each key process area is described in terms of the key practices that contribute to satisfying its goals. The ***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 referred to as the 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 sub practices. The key practices describe "what" is to be done, but they should not be interpreted as mandating "how" the goals should be achieved. Alternative practices may accomplish the goals of the key process area. The key practices should be interpreted rationally to judge whether the goals of the key process area are effectively, although perhaps differently, achieved. The key practices are contained in the "Key Practices of the Capability Maturity Model, Version 1.1" [Paulk93b], along with guidance on their interpretation.

4 Future Directions of the CMM

Achieving higher levels of software process maturity is incremental and requires a long-term commitment to continuous process improvement. Software organizations may take ten 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 U.S. companies, this level of effort is required to produce mature software organizations.

The CMM is not a silver bullet and does not address all of the issues that are important for successful projects. For example, the CMM 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, they have not been integrated into the CMM.

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 it is anticipated that CMM v1.1 will remain the baseline until at least 1996. This provides an appropriate and realistic balance between the needs for stability and for continued 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 (ISO) in its efforts to build international standards for software process assessment, improvement, and capability evaluation. This effort will integrate concepts from many different process improvement methods. The development of the ISO standards (and the contributions of other methods) will influence CMM v2.0, even as the SEI's process work will influence the activities of the ISO.

5 Conclusion

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.[Dion92, Humphrey91b, Lipke92, Wohlwend93]

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.

6 References

Dion92	Raymond Dion, "Elements of a Process-Improvement Program," IEEE Software, Vol. 9, No. 4, July 1992, pp. 83-85.
Fowler90	P. Fowler and S. Rifkin, *Software Engineering Process Group Guide*, Software Engineering Institute, CMU/SEI-90-TR-24, ADA235784, September, 1990.
Humphrey87a	W.S. Humphrey, *Characterizing the Software Process: A Maturity Framework*, Software Engineering Institute, CMU/SEI-87-TR-11, ADA182895, June 1987. Also published in IEEE Software, Vol. 5, No. 2, March 1988, pp.73-79.
Humphrey87b	W.S. Humphrey and W.L. Sweet, *A Method for Assessing the Software Engineering Capability of Contractors*, Software Engineering Institute, CMU/SEI-87-TR-23, ADA187320, September 1987.
Humphrey89	W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, Reading, MA, 1989.
Humphrey91a	W.S. Humphrey, D.H. Kitson, and J. Gale, "A Comparison of U.S. and Japanese Software Process Maturity," *Proceedings of the 13th International Conference on Software Engineering*, Austin, TX, 13-17 May 1991, pp. 38-49.
Humphrey91b	Watts S. Humphrey, T.R. Snyder, and Ronald R. Willis, "Software Process Improvement at Hughes Aircraft," IEEE Software, Vol. 8, No. 4, July 1991, pp. 11-23.
Kitson92	D.H. Kitson and S. Masters, *An Analysis of SEI Software Process Assessment Results: 1987-1991*, Software Engineering Institute, CMU/SEI-92-TR-24, July 1992.
Lipke92	W.H. Lipke and K.L. Butler, "Software Process Improvement: A Success Story," Crosstalk: The Journal of Defense Software Engineering, No. 38, November 1992, pp. 29-31.

Paulk91	M.C. Paulk, B. Curtis, M.B. Chrissis, et al, *Capability Maturity Model for Software*, Software Engineering Institute, CMU/SEI-91-TR-24, ADA240603, August 1991.
Paulk93a	M.C. Paulk, B. Curtis, M.B. Chrissis, and C.V. Weber, *Capability Maturity Model for Software, Version 1.1*, Software Engineering Institute, CMU/SEI-93-TR-24, February 1993.
Paulk93b	M.C. Paulk, C.V. Weber, S. Garcia, M.B. Chrissis, and M. Bush, *Key Practices of the Capability Maturity Model, Version 1.1*, Software Engineering Institute, CMU/SEI-93-TR-25, February 1993.
Weber91	C.V. Weber, M.C. Paulk, C.J. Wise, and J.V. Withey, *Key Practices of the Capability Maturity Model*, Software Engineering Institute, CMU/SEI-91-TR-25, ADA240604, August 1991.
Wohlwend93	H. Wohlwend and S. Rosenbaum, "Software Improvements in an International Company," *Proceedings of the 15th International Conference of Software Engineering*, Washington D.C, May 1993.

The Capability Maturity Model for Software was produced by a dedicated group of people who spent many hours discussing the model and its features and then trying to document it in the two versions of the CMM. Contributors to the CMM, other than the authors, 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 in our efforts. 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, Charlie Weber, and Ron Willis. Former members who worked with us on CMM v1.0 include Harry Carl, Jim Hess, Jerry Pixton, and Jim Withey.

This work was sponsored by the U.S. Department of Defense.

For Further Information

For further information regarding the CMM and its associated products, including training on the CMM and how to perform software process assessments and software capability evaluations, contact:

SEI Customer Relations
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-5800
Internet: customer-relations@sei.cmu.edu