1 The Capability Maturity Model[SM] for Software[†]

Mark C. Paulk, Charles V. Weber, and Mary Beth Chrissis
Software Engineering Institute, USA

This chapter provides an overview of the Capability Maturity Model for Software (CMM V1.1). CMM V1.1 describes the software engineering and management practices that characterize organizations as their processes for developing and maintaining software mature. This chapter stresses the need for a process maturity framework to prioritize improvement actions; describes the five maturity levels, key process areas, and their common features; and discusses future directions for the CMM.

Introduction

In many organizations, software projects are often late and over budget. This state of affairs is sometimes referred to as the "software crisis." In 1986 the Software Engineering Institute (SEI), with assistance from the MITRE Corporation, began developing a process maturity framework that would help organizations improve their software process; this has evolved into the Capability Maturity Model for Software (CMM or SW-CMM[1]) [11, 6, 12].

The SW-CMM presents sets of recommended practices in a number of key process areas that have been shown to enhance software process capability. It 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. By focusing on a limited set of issues and working aggressively to address them, an organization can steadily improve its organization-wide software process to enable continual and lasting gains in software

[SM] CMM, Capability Maturity Model, and IDEAL are service marks of Carnegie Mellon University.

† The U.S. Department of Defense sponsored this work.

[1] Several CMMs inspired by the CMM for Software have now been developed. To minimize confusion, we use SW-CMM to identify the original CMM for Software.

process capability.

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.

In contrast, 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 mandated processes 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.

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 work products (for instance, 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. An organization's software process capability is one way of predicting the most likely outcome to expect 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 an organizational culture that support the methods, practices, and procedures of the business so that they endure even after those who originally defined them have gone.

The Five Levels of Software Process Maturity

Continual process improvement is based on many small, evolutionary steps, although revolutionary innovations may be part of a process improvement program. The staged structure of the SW-CMM is based on principles of product quality espoused by Walter Shewart, W. Edwards Deming, Joseph Juran, and Philip Crosby. The SW-CMM provides a framework for organizing these evolutionary steps into five maturity levels that lay successive foundations for continual 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 higher level of process capability for the organization.

Organizing the SW-CMM into the five levels shown in Figure 1-1 prioritizes improvement actions for increasing software process capability. The labeled arrows in Figure 1-1 indicate the type of process capability being institutionalized by the organization at each step of the maturity framework.

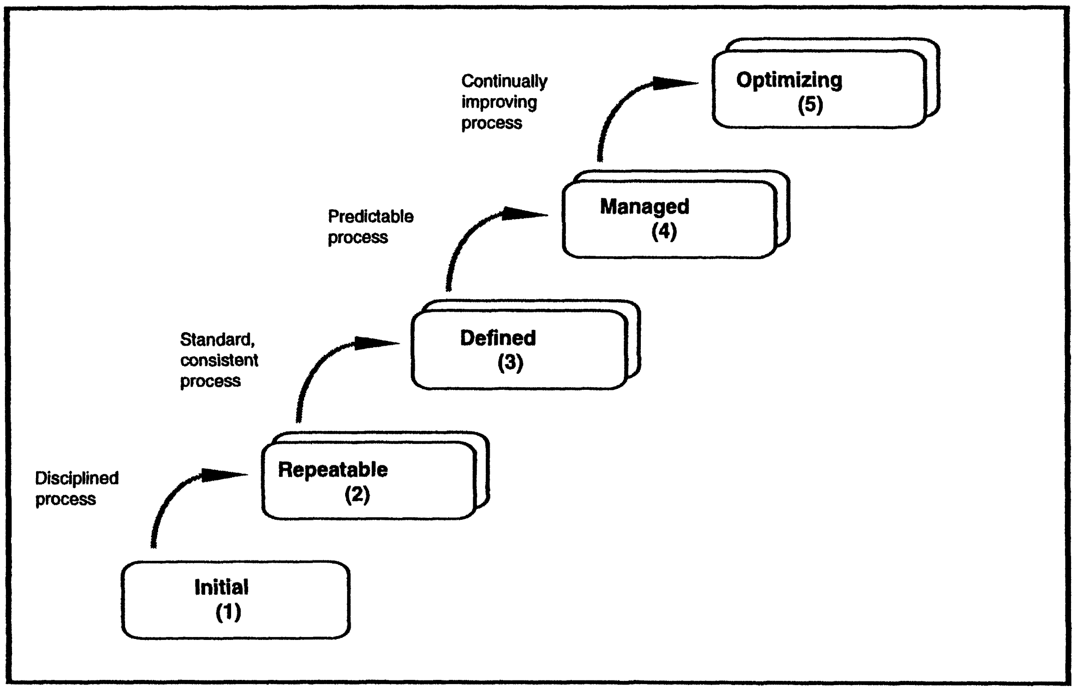


Figure 1-1: The five levels of software process maturity.

The five levels can be briefly described as:

1) Initial	The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
2) Repeatable	Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3) Defined	The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. Projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.
4) Managed	Detailed measures of the software process and product quality are collected, analyzed, and used to control the process. Both the software process and products are quantitatively understood and controlled.
5) Optimizing	Continual process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

These five levels reflect the fact that the SW-CMM is a model for improving the capability of software organizations. The priorities in the SW-CMM, as expressed by these levels, are not directed at individual projects. A project that is in trouble might justifiably set its priorities for corrective action differently from those in the SW-CMM. Its solutions, however, might be of limited value to the rest of the organization, because other projects might have different problems or because other projects lack the necessary foundation to take advantage of its solutions. The SW-CMM focuses on processes that build organizational capability.

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.

Level 1 - The Initial Level

At the Initial Level, the organization typically does not provide a stable environment for developing and maintaining software. Over-commitment is a characteristic of Level 1 organizations, and 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 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 exceed the budget and schedule. Success in Level 1 organizations depends on the competence and heroics of the people in the organization[2] 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.

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 are

[2] 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 SW-CMM.

based on experience with similar projects. Establishing basic process management discipline on a project-by-project basis enhances process capability. Projects implement effective processes that are defined, documented, practiced, trained, measured, enforced, and provide a basis for improvement.

Projects in Level 2 organizations have installed basic software management controls. Realistic project commitments are made, 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, and their integrity is controlled. Software project standards are defined, and the organization ensures that they are faithfully followed. The software project works with its subcontractors, if any, to establish an effective customer-supplier relationship.

Processes may differ among 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 software project planning and tracking are 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.

Level 3 - The Defined Level

At the Defined Level, a standard process (or processes) for developing and maintaining software is documented and used across the organization. This standard process includes both software engineering and management processes integrated into a coherent whole. This standard process is referred to throughout the SW-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. A group such as a software engineering process group or SEPG is responsible for the organization's software process activities. An organization-wide training program is implemented to ensure that the staff and managers have the knowledge and skills required to perform 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 SW-CMM as the *project's defined software process*. It is the process used in performing the project's activities. A defined software process contains a coherent, integrated set of well-defined software engineering and management processes. A well-defined process includes entry criteria, inputs, standards and procedures for performing the work, verification mechanisms (such as peer reviews), outputs, and exit criteria. Because the software process is well defined, management has good insight into technical progress on the project.

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.

Level 4 - The Managed Level

At the Managed Level, the organization sets quantitative quality goals for both software products and processes. Productivity, quality, etc. 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. 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 "quantified and predictable" because the process is measured and operates within quantitative limits. This level of process capability allows an organization to predict trends in process performance 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 pre-defined limits are exceeded, actions are taken to understand and correct the situation. Software products are of predictably high quality.

Level 5 - The Optimizing Level

At the Optimizing Level, the entire organization is focused on continual process improvement. The organization has the means to identify weaknesses and strengthen the process proactively, with the goals of preventing defects and improving efficiency. Data on process effectiveness are 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 teams in Level 5 organizations analyze defects to determine their causes, evaluate software processes to prevent known types of defects from recurring, and disseminate lessons learned throughout the organization.

There is chronic waste, in the form of rework, in any system simply because of random variation. Organized efforts to remove waste result in changing the system by addressing "common causes" of inefficiency. While efforts to reduce waste occur at all maturity levels, it is the focus of Level 5.

The software process capability of Level 5 organizations can be characterized as "continually improving" because Level 5 organizations are continually striving to improve the range of their process capability, thereby improving the process performance of their projects. Improvements occur 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.

Process Capability and the Prediction of Performance

An organization's software process maturity helps to predict a project's ability to meet its objectives. Projects in Level 1 organizations experience wide variations in achieving cost, schedule, functionality, and quality targets. Figure 1-2 illustrates the kinds of improvements expected in predictability, control, and effectiveness in the form of a probability density for the likely performance of a particular project with respect to targets, such as cycle time, cost, and quality.

The first improvement expected as an organization matures is in predictability. 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.

The second improvement is in control. As maturity increases, the variability of actual results around targeted results decreases. For instance, in Level 1 organization 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.

The third improvement is in effectiveness. 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 effectiveness and reduced costly rework, allowing development time to be shortened.

The improvements in predicting a project's results represented in Figure 1-2 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 to identify and address problems earlier than for 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.

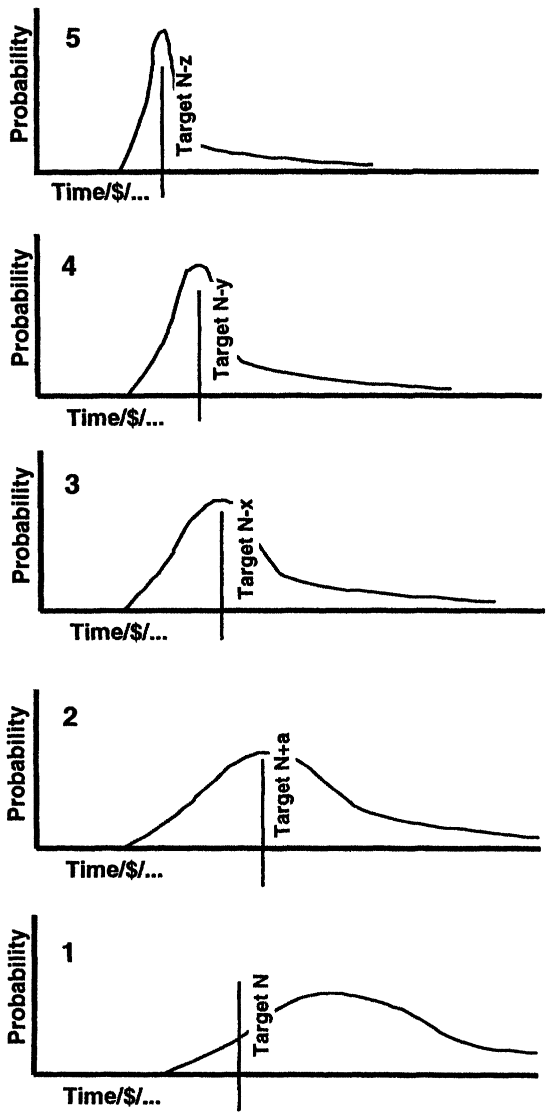
The documented case studies of software process improvement indicate that there are significant improvements in both quality and productivity as a result of the improvement effort [5, 9]. The return on investment seems to typically be in the 4:1 to 8:1 range for successful process improvement efforts, with increases in productivity ranging from 9-67 percent and decreases in cycle time ranging from 15-23 percent reported [5].

Summarizing the Key Process Areas

The SW-CMM is a framework representing a path of improvements recommended for software organizations that want to increase their software process capability. The intent is that the SW-CMM is at a sufficient level of abstraction that it does not unduly constrain how the software process is implemented by an organization. The SW-CMM describes what we would normally expect in a software process, regardless of how the process is implemented.

Each maturity level, with the exception of Level 1, has been decomposed into constituent parts. The decomposition of each maturity level ranges from abstract summaries of each level down to their operational definition in the key practices. Each maturity level is composed of several key process areas, which indicate where an organization should focus to improve its software process. 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. The common features are commitment to perform, ability to perform, activities performed, measurement and analysis, and verifying implementation.

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 1-3. 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.



Performance continually 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 1-2: Process capability as indicated by maturity level.

Level	Focus	Key Process Areas
5 Optimizing	*Continual process improvement*	Defect Prevention Technology Change Management Process Change Management
4 Managed	*Product and process quality*	Quantitative Process Management Software Quality Management
3 Defined	*Engineering processes and organizational support*	Organization Process Focus Organization Process Definition Training Program Integrated Software Management Software Product Engineering Intergroup Coordination Peer Reviews
2 Repeatable	*Project management processes*	Requirements Management Software Project Planning Software Project Tracking & Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management
1 Initial	*Competent people and heroics*	

Figure 1-3: The key process areas by maturity level.

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

The key process areas are the "requirements" for achieving a maturity level. To achieve a maturity level, the key process areas for that level and the lower levels must be satisfied (or not applicable, such as *Software Subcontract Management* when there are no subcontractors).

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 evolve to take advantage of the organization's software process assets available at Level 3, as 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:

- *Requirements Management*: 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.
- *Software Project Planning*: 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.
- *Software Project Tracking and Oversight*: 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.
- *Software Subcontract Management*: Select qualified software subcontractors and manage them effectively.
- *Software Quality Assurance*: Provide management with appropriate visibility into the process being used by the software project and of the products being built.
- *Software Configuration Management*: 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:

- *Organization Process Focus*: Establish the organizational responsibility for software process activities that improve the organization's overall software process capability.
- *Organization Process Definition*: Develop and maintain a usable set of software process assets that improve process performance across the projects and provides 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.
- *Training Program*: Develop the skills and knowledge of individuals so that 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.

- *Integrated Software Management*: 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.
- *Software Product Engineering*: 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, for instance, requirements analysis, design, code, and test.
- *Intergroup Coordination*: 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.
- *Peer Reviews*: 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:

- *Quantitative Process Management*: Control 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.
- *Software Quality Management*: 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 continual and measurable software process improvement:

- *Defect Prevention*: Identify the causes of defects and prevent them from recurring. The software project analyzes defects, identifies their causes, and changes its defined software process.
- *Technology Change Management*: Identify beneficial new technologies (such as tools, methods, and processes) and transfer them into the or-

ganization in an orderly manner. The focus of Technology Change Management is on performing innovation efficiently in an ever-changing world.

- *Process Change Management*: Continually improves the software processes used in the organization with the intent of improving software quality, increasing productivity, and decreasing the cycle time for product development.

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 key process area is determined by achievement of the goals.

Key practices describe the activities and infrastructure that contribute most to the effective implementation and institutionalization of the key process area. Each key practice consists of a single sentence, usually 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 subpractices. 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 IDEAL Approach to Software Process Improvement

Effective software process improvement, whether based on the SW-CMM or some other model, occurs in a systematic fashion [14]. The SEI has developed the IDEAL model, shown in Figure 1-4, to depict the activities of an improvement program based on the SW-CMM [10]. It consists of five phases:

I	Initiating (the improvement program)
D	Diagnosing (the current state of practice)
E	Establishing (the plans for the improvement program)
A	Acting (on the plans and recommended improvements)
L	Leveraging (the lessons learned and the business results of the improvement effort)

The *Initiating* phase establishes the business reasons for undertaking a software process improvement effort. It identifies high-level concerns in the organization that can be the stimulus for addressing various aspects of quality improvement. Communication of these concerns and business perspectives is needed during the Initiating phase in order to gain visible executive buy-in and sponsorship at this very early part of the improvement effort.

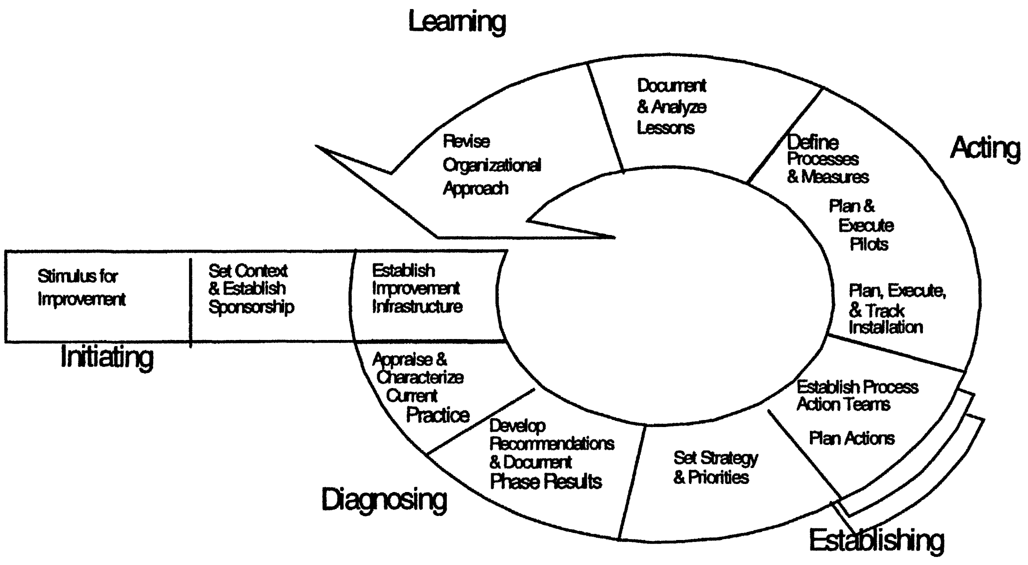

Learning
Document & Analyze Lessons
Revise Organizational Approach
Define Processes & Measures
Acting
Plan & Execute Pilots
Plan, Execute, & Track Installation
Stimulus for Improvement
Set Context & Establish Sponsorship
Establish Improvement Infrastructure
Initiating
Appraise & Characterize Current Practice
Develop Recommendations & Document Phase Results
Diagnosing
Set Strategy & Priorities
Establish Process Action Teams
Plan Actions
Establishing

Figure 1-4: The SEI's IDEAL approach to software process improvement.

The *Diagnosing* phase is used to build a common understanding of the current processes of the organization, especially the strengths and weaknesses of those current processes. It will also help identify priorities for improving your software processes. This diagnosis is based on the SW-CMM (or one of the other CMMs).

The *Establishing* phase finalizes the strategy and supporting plans for the software process improvement program. It sets the direction and guidance for the next three to five years, including strategic and tactical plans for software process improvement.

The *Acting* phase takes action to effect changes in organizational systems that result in improvements in these systems. These improvements are made in an orderly manner and in ways that will cause them to be sustained over time. Techniques used to support and institutionalize change include defining software processes and measurements, pilot testing, and installing new processes and measurements throughout the organization. The SW-CMM provides guidance for the improvement actions in its informative components: the key practices and their subpractices and examples.

The *Leveraging* phase completes the process improvement cycle. Lessons learned from the pilot projects and improvement efforts are documented and analyzed in order to improve the process improvement program for the future. The business needs that were determined at the beginning of the cycle are revisited to see if they have been met. Sponsorship for the program is revisited and renewed for the next software process improvement cycle.

The Continuing Evolution of the SW-CMM

The SW-CMM is a living document; Version 2 is currently under development. A number of major changes were proposed for SW-CMM V2 [15].

Architecture and Templates

We received change requests:

- for key process areas to span maturity levels
- to incorporate a description of process evolution
- to provide a finer granularity of process rating

As part of our participation in SPICE (see Chapter 3), we investigated alternatives that would systematically address these requests [7, 13]. The SW-CMM V2 focuses on the *vital few* issues that enable process improvement with its *staged* architecture. The *continuous* architecture used by SPICE and the Systems Engineering CMM [1] provides another useful perspective. The continuous perspective describes the evolution of processes and provides greater flexibility and finer granularity in rating processes. Both the staged and continuous perspectives have value, and they are conceptually compatible.

Our solution is to make the relationships between the two perspectives explicit. SW-CMM V2 will remain a *staged* model, but it will also explicitly state relationships to the *continuous* architecture, which will be published as an appendix. Templates for the key process areas will directly correspond to generic practices in the continuous architecture.

Version 2 will use templates more extensively and consistently than Version 1. The templates will systematically change as appropriate at each maturity level to capture the levels' institutionalization.

A goal will be added to each key process area to capture the institutionalization of the process. This goal will be stated as *<Do X> according to a <maturity level> process*. The *<maturity level> process* captures institutionalization concepts such as documenting, training, tailoring, etc., as appropriate for the maturity level. For example, "perform software project planning according to a repeatable process" will capture planning, polices, resources, responsibilities, training, etc., that are part of institutionalizing a repeatable process for planning.

This plan to add a goal to each key process area has the following advantages:

- It clarifies that institutionalization is a critical part of achieving a key process area. The key practices in the institutionalization common features (commitment, ability, measurement, verification) will map directly to this goal.

- It separates institutionalization and implementation for purposes of rating key process areas. The separation of concerns simplifies some appraisal rating decisions.
- It supports the strategy of mapping to a continuous perspective of software process maturity. Processes evolve, even though key process areas reside at a single maturity level.

Active Versus Passive Voice

Active voice is easier to read and understand than passive voice. However, the use of active voice may imply that a key process area is a collection of steps in a process. A key process area describes some of the crucial attributes of what a process is, not how the process should be implemented. Practices will be rewritten in active voice. This means changing every key practice in V1.1, even though this change will not materially affect the intent.

Supplier Management

At Level 2, *Software Subcontract Management* will be expanded to include off-the-shelf and customer-supplied software. The new name of the key process area will be *Software Supplier Management.*

Risk Management

Users have suggested that risk management is a vital element of modern project management that is inadequately addressed in the current SW-CMM V1.1. This was the most controversial change proposed for Version 2. At Level 3, the SEI prototyped a *Software Risk Management* key process area as part of our leadership role in extending software process improvement and in updating the SW-CMM to reflect current best practice. The final decision, however, was to add a goal on risk management in *Integrated Software Management.*

Product Line Engineering

Users have suggested that reuse, while it may be irrelevant for some organizations, is an important advancement in the software industry. Research also suggests a strong correlation between higher maturity levels and systematic reuse (or product lines) [2]. Product line engineering, systematic reuse, reengineering, and aligning with strategic business goals will be addressed at Level 4.

Split Quantitative Process Management

Quantitative Process Management will be split into two key process areas at Level 4. *Statistical Process Management*, will address quantitative process control. The other key process area, *Organization Process Performance*, will focus on the organizational aspects of establishing quantitative expectations for process capability.

Integrating CMMs

The success of the Software CMM has inspired the development of a number of other capability maturity models [8], addressing topics such as systems engineering [1], people issues [3], and software acquisition [4]. Integrating these models effectively and efficiently during appraisals and for process improvement can be challenging. SW-CMM v2 will satisfy a set of CMM integration criteria planned for release in August 1997.

Version 2 Plans

The software community is reviewing drafts of Version 2, and the Software CMM Advisory Board and Change Control Board will consider whether these changes are appropriate for release in Version 2.0.

Information on the drafts of SW-CMM v2 and related work is available on the CMM Correspondence Group Web page:

- *http://www.sei.cmu.edu/technology/CMM/cg.html*

and the SW-CMM v2 Web page:

- *http://www.sei.cmu.edu/technology/CMM/CMM.v2.html*

Version 2.0 and Version 1.1 of the Software CMM will both be supported during a transition period that will end with the release of Version 2.1, a minor upgrade planned for 1999.

Conclusion

To quote George Box, "All models are wrong; some models are useful." The SW-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 SW-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 SW-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 its use improves the likelihood that a software organization will achieve its cost, quality, and productivity goals.

For further information regarding the SW-CMM and its associated products, including training on the SW-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

The SEI Web page is:

http://www.sei.cmu.edu/.

For information specifically on the SW-CMM, see:

http://www.sei.cmu.edu/technology/CMM.html.

References

1. Bate, R., Kuhn, D., Wells, C., et al., *A Systems Engineering Capability Maturity Model,* Version 1.1, CMU/SEI-95-MM-003, Carnegie Mellon University, Software Engineering Institute, November 1995.

2. Besselman, J. and Rifkin, S., "Exploiting the Synergism Between Product Line Focus and Software Maturity," *Proceedings of the 1995 Acquisition Research Symposium*, Washington, D.C., pp. 95-107.

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

4. Ferguson, J., Cooper, J., et al., *Software Acquisition Capability Maturity Model (SA-CMM),* Version 1.01, CMU/SEI-96-TR-020, Carnegie Mellon University, Software Engineering Institute, December 1996.

5. Herbsleb, J., Carleton, A., et al., *Benefits of CMM-Based Software Process Improvement: Initial Results,* CMU/SEI-94-TR-13, Carnegie Mellon University, Software Engineering Institute, August 1994.

6. Humphrey, W. S., *Managing the Software Process*, Addison-Wesley, Reading, MA, 1989.

7. Konrad, M. D., Paulk, M. C., and Graydon, A. W., "An Overview of SPICE's Model for Process Management," *Proceedings of the Fifth International Conference on Software Quality,* Austin, TX, 23-26 October 1995, pp. 291-301.

8. Konrad, M., Chrissis, M.B., Ferguson, J., Garcia, S., Hefley, B., Kitson, D., and

Paulk, M., "Capability Maturity Modeling at the SEI," *Software Process: Improvement and Practice*, Vol. 2, Issue 1, March 1996, pp. 21-34.

9. Lawlis, P. K., Flowe, R. M., and Thordahl, J. B., "A Correlational Study of the CMM and Software Development Performance, *Crosstalk: The Journal of Defense Software Engineering*, Vol. 8, No. 9, September 1995, pp. 21-25.

10. McFeeley, B., *IDEAL: A User's Guide for Software Process Improvement*, CMU/SEI-96-HB-001, Carnegie Mellon University, Software Engineering Institute, February 1996.

11. Carnegie Mellon University, Software Engineering Institute (Principal Contributors and Editors: Paulk, M.C., Weber, C.V., Curtis, B., and Chrissis, M.B.), *The Capability Maturity Model: Guidelines for Improving the Software Process*, ISBN 0-201-54664-7, Addison-Wesley Publishing Company, Reading, MA, 1995.

12. Paulk, M.C., "The Evolution of the SEI's Capability Maturity Model for Software," *Software Process: Improvement and Practice*, Vol. 1, Pilot Issue, Spring 1995, pp. 3-15.

13. Paulk, M.C., Konrad, M.D., and Garcia, S.M., "CMM Versus SPICE Architectures," *Software Process Newsletter*, IEEE Computer Society Technical Council on Software Engineering, No. 3, Spring 1995, pp. 7-11.

14. Paulk, M. C., "Effective CMM-Based Process Improvement," *Proceedings of the 6th International Conference on Software Quality*, Ottawa, Canada, 28-31 October 1996, pp. 226-237.

15. Paulk, M.C., Garcia, S.M., Chrissis, M.B., and Hayes, W., "SW-CMM V2: Feedback on Proposed Changes," *Software Process Newsletter*, IEEE Computer Society Technical Council on Software Engineering, No. 7, Fall 1996, pp. 5-10.