Interpreting the CMM

However, does not mean that the process is efficient in achieving its purpose. There may be many factors influencing both organization and project success. For example, a successful project that builds a product that no one buys is a failure in the commercial world.

"Goodness" attributes can only be interpreted in the context of the business environment and specific circumstances of the project and the organization. Such "goodness" judgments can be made only by the organization as part of its continuous process improvement cycle. Perfection is never achieved, and continuous process improvement never ends.
Interpreting the CMM

What are the criteria for a "reasonable" software process? A reasonable software process is one that is effective in building the organizational capability and satisfies most of the requirements of a defined process. Specifically, it is practiced, documented, enforced, trained, measured, and able to improve.

If an organization established a software process for estimating that consisted of rolling the dice, would that constitute a reasonable process? It could certainly be documented and consistently followed. Some might even argue that it would be as realistic as many estimating techniques. "Rolling the dice" would, however, not be judged a reasonable estimating process by most software professionals. Since it responds only to the laws of probability, it cannot be improved.

How far is it from "rolling the dice" to documenting a process to "go ask George?" This could be a very good method for estimating. As long as George is around, it could even be consistent and repeatable. It would not, however, satisfy our criteria since it cannot be trained to other individuals. It is a person-centered process that cannot be repeated without George. It does not build an ongoing organizational capability.

Using some variant of a Delphi method (a method where experts in a subject review the issues under consideration and come to consensus on the recommendations related to the issue) for estimating would usually be judged a reasonable software process. A size estimating approach based on a Delphi method satisfies the criteria for a reasonable and effective process, even though the Delphi method is a person-centered process. An organizational capability can be based on a structured technique such as a Delphi method.

In a fundamental sense, professional judgment is necessary to make such distinctions. The difficulty lies in discriminating between compliance and goodness. The goals summarize the key practices, which, in turn, describe a reasonable software process. Complying with a reasonable process,
Interpreting the CMM

examples provided in many cases, at least for the first use of the term. These phrases may have different meanings for two different organizations, for two projects in a single organization, or for one project at different points in its life cycle. Each project or organization must clarify these phrases for its specific situation.

Clarifying these phrases requires the organization to consider the overall context in which they are used. The pertinent question is whether the specific interpretation of one of these phrases meets the goals of the key process area. Professional judgment must be used to determine whether the goals have been achieved. The glossary in Appendix B may provide guidance in interpreting these and other phrases in the key practices.

Professional judgment must also be used when interpreting the key practices and how they contribute to the goals of a key process area. In general, the key process areas describe a fundamental set of behaviors that all software organizations should exhibit, regardless of their size or their products. The key practices in the CMM, however, must be interpreted in light of a project's or organization's business environment and specific circumstances. This interpretation should be based on an informed knowledge of both the CMM and the organization and its projects. The goals of the key process areas provide a means for structuring this interpretation. If an organization's implementation of a key process area satisfies the goals, but differs significantly from the key practices, the rationale for the interpretation should be documented. A documented rationale will help assessment and evaluation teams understand why certain practices are implemented the way they are.

Applying professional judgment leads to the issue of the "goodness" of the software process. The CMM does not place "goodness" requirements on the software process, although it does establish minimal criteria for a "reasonable" process in many software environments. The objective of process management is to establish processes that are used and can act as a foundation for systematic improvement based on the organization's business needs.
Interpreting the CMM

pressures. Ensuring effective operational independence without the organizational independence is difficult. For example, an employee reporting to the project manager may be reluctant to stop a test activity even though serious noncompliance issues exist.

Organizations must determine the organizational structure that will support activities that require independence, such as SQA, in the context of their strategic business goals and business environment.

Independence should:

- provide the individuals performing the SQA role with the organizational freedom to be the "eyes and ears" of senior management on the project,
- protect the individuals performing the SQA role from performance appraisal by the management of the project about which they are reporting, and
- provide senior management with confidence that objective information on the process and products of the project is being reported.

Since the key practices allow interpretation of the independence criteria, professional judgment must be exercised by the organization in determining whether the goals of the key process area are achieved.

4.5 Applying Professional Judgment

To provide a complete set of valid principles that apply to a wide range of situations, some of the key practices are intentionally stated to allow for flexibility. Throughout the key practices, nonspecific phrases like "affected groups," "as appropriate," and "as necessary" are used. The use of such nonspecific terms is generally minimized in the key practices, with
Interpreting the CMM

Training group

The training group is the collection of individuals (both managers and staff) who are responsible for coordinating and arranging the training activities for an organization. This group typically prepares and conducts most of the training courses and coordinates use of other training vehicles.

4.4.3 Independence and Organizational Structure

The organization must take care that the key practices that call for independence are appropriately interpreted and followed. This is particularly true for small projects and small organizations. The key practices call for independence when technical or organizational biases may affect the quality or risks associated with the project. For example, two practices dealing with independence are:

- The SQA group has a reporting channel to senior management that is independent of the project manager, the project’s software engineering group, and the other software-related groups (Commitment 1.2 in Software Quality Assurance).
- The (system and acceptance) test cases and test procedures are planned and prepared by a test group that is independent of the software developers (Activity 7.3 in Software Product Engineering).

The need for independence of the system and acceptance testing is based on technical considerations. This independence ensures that the testers are not inappropriately influenced by the design and implementation decisions made by the software developers or maintainers.

The independence of the SQA group is necessary so its members can perform their jobs without being influenced by project schedule and cost...
Interpreting the CMM

**System engineering group**
The system engineering group is the collection of individuals (both managers and technical staff) who have responsibility for specifying the system requirements; allocating the system requirements to the hardware, software, and other components; specifying the interfaces between the hardware, software, and other components; and monitoring the design and development of these components to ensure conformance with their specifications.

**System test group**
The system test group is the collection of individuals (both managers and technical staff) who have responsibility for planning and performing the independent system testing of the software to determine whether the software product satisfies its requirements.

**Software quality assurance group**
The software quality assurance group is the collection of individuals (both managers and technical staff) who plan and implement the project’s quality assurance activities to ensure the software process steps and standards are followed. Organizational issues concerning software quality assurance are discussed in Section 4.4.3.

**Software configuration management group**
The software configuration management group is the collection of individuals (both managers and technical staff) who have responsibility for planning, coordinating, and implementing the formal configuration management activities for the software project.
Groups commonly referred to in the CMM are described below:

**Software engineering group**

The software engineering group is the collection of individuals (both managers and technical staff) who have responsibility for software development and maintenance activities (i.e., requirements analysis, design, code, and test) for a project.

Groups performing software-related work, such as the software quality assurance group, the software configuration management group, and the software engineering process group, are not included in the software engineering group. These groups are considered to be one of the "other software-related groups."

**Software-related groups**

A software-related group is the collection of individuals (both managers and technical staff) representing a software engineering discipline that supports, but is not directly responsible for, software development and/or maintenance.

Examples of software engineering disciplines include software quality assurance and software configuration management.

**Software engineering process group**

The software engineering process group is the group of specialists who facilitate the definition, maintenance, and improvement of the software process used by the organization. In the key practices, this group is generically referred to as "the group responsible for the organization’s software process activities."
### Interpreting the CMM

**Organization**

An organization is a unit within a company or other entity (e.g., government agency or branch of service) within which many projects are managed as a whole. All projects within an organization share a common top-level manager and common policies.

**Project**

A project is an undertaking requiring concerted effort, which is focused on developing and/or maintaining a specific product. The product may include hardware, software, and other components. Typically a project has its own funding, cost accounting, and delivery schedule.

**Group**

A group is the collection of departments, managers, and individuals who have responsibility for a set of tasks or activities. A group could vary from a single individual assigned part time, to several part-time individuals assigned from different departments, to several individuals dedicated full time.
Interpreting the CMM

In a particular project or organization, there does not need to be a one-to-one correspondence between these roles and individuals. One person could perform in multiple roles, or each role could be performed by separate individuals.

For example, on a small, software-only project, one person might have as many as six roles: the system engineering first-line manager, the project system engineering manager, the software first-line manager, the project software manager, the project manager, and the software configuration management manager.

On a slightly larger project, one person might be the system engineering first-line manager, the project system engineering manager, and the project manager while another person might be both the first-line software manager and the project software manager. These two managers might be in the same second-line organization or in different second-line organizations.

On a large project, many roles, especially those of management, would likely be filled by separate individuals.

4.4.2 Organizational Structure

The fundamental concepts of organization, project, and group must be understood to properly interpret the key practices of the Capability Maturity Model. The following paragraphs define the use of these concepts in the CMM:
Interpreting the CMM

Software task leader  A software task leader fulfills the role of leader of a technical team for a specific task, who has technical responsibility and provides technical direction to the staff working on the task.

The software task leader usually reports to the same first-line software manager as the other people who are working on the task.

Staff, software engineering staff, individuals

Several terms are used in the CMM to denote the individuals who perform the various technical roles described in various key practices of the CMM. The staff are the individuals, including task leaders, who are responsible for accomplishing an assigned function, such as software development or software configuration management, but who are not managers.

The software engineering staff are the software technical people (e.g., analysts, programmers, and engineers), including software task leaders, who perform the software development and maintenance activities for the project, but who are not managers.

The term "individuals" as used in the key practices is qualified and bounded by the context in which the term appears (e.g., "the individual involved in managing the software subcontract").

A similar breakout of roles can be identified for other engineering groups such as system engineering or system test.
Interpreting the CMM

Project software manager

A project software manager fulfills the role with total responsibility for all the software activities for a project. The project software manager is the individual the project manager deals with in terms of software commitments and who controls all the software resources for a project.

The software engineering groups on a project would report to the project software manager, although some activities such as tools development might have a matrixed reporting relationship.

In a large project, the project software manager is likely to be a second-, third-, or fourth-line manager. In a small project or department with a single project, the project software manager might be the first-line software manager or might be at a higher level.

First-line software manager

A first-line software manager fulfills the role with direct management responsibility (including providing technical direction and administering the personnel and salary functions) for the staffing and activities of a single organizational unit (e.g., a department or project team) of software engineers and other related staff.
Interpreting the CMM

Project manager

A project manager fulfills the role with total business responsibility for an entire project; the project manager is the individual who directs, controls, administers, and regulates a project building a software or hardware/software system. The project manager is the individual ultimately responsible to the customer.

In a project-oriented organizational structure, most of the people working on a project would report to the project manager, although some disciplines might have a matrixed reporting relationship. In a matrixed organizational structure, it may be only the business staff who reports to the project manager. The engineering groups would then have a matrixed reporting relationship.
A senior manager fulfills a management role at a high enough level in an organization that the primary focus is the long-term vitality of the organization, rather than short-term project and contractual concerns and pressures. In general, a senior manager for engineering would have responsibility for multiple projects. A senior manager also provides and protects resources for long-term improvement of the software process (e.g., a software engineering process group).

Senior management, as used in the CMM, can denote any manager who satisfies the above description, up to and including the head of the whole organization. As used in the key practices, the term senior management should be interpreted in the context of the key process area and the projects and organization under consideration. The intent is to include specifically those senior managers who are needed to fulfill the leadership and oversight roles essential to achieving the goals of the key process area.
4.4 Organizational Structure and Roles

Although the CMM attempts to remain independent of specific organizational structures and models, it is necessary to express the practices in the CMM consistently using terminology related to organizational structure and roles, which may differ from that followed by any specific organization. The following sections describe the various concepts related to organizations, projects, and roles that are necessary for interpreting the key practices of the CMM.

4.4.1 Organizational Roles

A role is a unit of defined responsibilities that may be assumed by one or more individuals. The following descriptions of roles are frequently used in the key practices:

**Manager**

A manager fulfills a role that encompasses providing technical and administrative direction and control to individuals performing tasks or activities within the manager’s area of responsibility. The traditional functions of a manager include planning, resourcing, organizing, directing, and controlling work within an area of responsibility.
4.3.8 Collection and Analysis of Process Data

The key practices for the collection and analysis of process data evolve across the maturity levels.

At Level 2, the data are primarily related to the size of the project's work products, effort, and schedule, and are defined, collected, and stored separately by each project. The data are shared between projects via informal mechanisms.

At Level 3, each project has a defined software process tailored from the organization's standard software process. Data related to each project's defined software process are collected and stored in the organization's software process database. The data collected and stored may be different for each project, but the data are well defined within the organization's software process database.

At Level 4, the organization defines a standard set of measurements based on the organization's standard software process. All projects collect this standard set of measurement data, as well as other project-specific data, and store them in the organization's software process database. The data are used by the projects to quantitatively understand and stabilize the process performance of the projects' defined software processes. They are also used by the organization to establish a process capability baseline for the organization's standard software process.

At Level 5, data are used to select areas for technology and process improvements, plan these improvements, and evaluate the effects of these improvements on the organization's process capability.
4.3.6 Technology and the CMM

The key practices neither require nor preclude specific software technologies, such as prototyping, object oriented design, or reusing software requirements, design, code, or other elements.

4.3.7 Documentation and the CMM

The key practices describe a number of process-related documents, each one covering specific areas of content. The key practices do not require a one-to-one relationship between the documents named in the key practices and the actual work products of an organization or project. Nor is there an intended one-to-one relationship to documents specified by the DoD or to standards such as DOD-STD-2167A or IEEE software standards. The key practices require only that the applicable contents of these documents be part of the organization's or project's written work products.

In terms of document structure, the contents of a document referred to in the key practices could be part of a larger document. For example, an organization might have a software development plan that includes the essentials of the software risk management plan.

Alternatively, the contents of a document referred to in the key practices could be distributed over a set of documents that differ from the set named in the key practices. For example, a project might develop three documents, a software development plan, a software management plan, and a project work breakdown structure, to satisfy the key practices for a software project's software risk management, software quality assurance, and software development plans.
4.3.4 Relationship Between the Project's Defined Software Process and the Software Development Plan

The description of the project's defined software process will usually not be specific enough to be performed directly. Although the description typically identifies such things as roles (i.e., who performs a task) and types of software work products needed to perform a task, it does not specify the individual who will assume the roles, the specific software work products that will be created, nor the schedule for performing the tasks and activities.

The project's software development plan, either as a single document or a collection of plans collectively referred to as a software development plan, provides the bridge between the project's defined software process (what will be done and how it will be done) and the specifics of how the project will be performed (e.g., which individuals will produce which software work products according to what schedule). The combination of the project's defined software process and its software development plan makes it possible to actually perform the process.

4.3.5 Life Cycles and the CMM

The key practices are not meant to limit the choice of a software life cycle. People who have extensively used one particular software life cycle may perceive elements of that life cycle in the organization and structure of the key practices. However, there is no intent either to encourage or preclude the use of any particular software life cycle.

The term "stage" is used to refer to a defined partition of the software project's effort, but the term should not be tied to any specific software life cycle. As it is used in the key practices, "stage" can mean rigidly sequential stages or overlapping and iterative stages.
Interpreting the CMM

Software work products (project results)

The results of activities and tasks primarily consist of software work products. A software work product is any artifact created as part of defining, maintaining, or using a software process, including process descriptions, plans, procedures, computer programs, and associated documentation, which may or may not be intended for delivery to a customer or end user. Work products become an input to the next step in the process or provide archival information on the software project for use in future projects.

Examples of software work products include plans, estimates, data on actual effort, corrective action documentation, and requirements documents. The subset of software work products that are deliverable to the customer or end user are referred to as software products.

Software products

The software products are the complete set, or any of the individual items of the set, of computer programs, procedures, and associated documentation and data designated for delivery to a customer or end user. [IEEE-STD-610]

All software products are also software work products. A software work product that will not be delivered to a customer or end use is not, however, a software product.
**Tasks**

The work to be performed is broken down into tasks. A task is a well-defined unit of work in the software process that provides management with a visible checkpoint into the status of the project. Tasks have readiness criteria (preconditions) and completion criteria (postconditions).

Within the context of process definition, a task is a well-defined component of a defined process. All tasks can be considered activities, but not all activities are well enough defined to be considered tasks (although an activity may include a task). Because of this, use of "task" in the Level 2 key practices is avoided and the less rigorous term "activity" is used.

**Activities**

An activity is any step taken or function performed, both mental and physical, toward achieving some objective. Activities include all the work the managers and technical staff do to perform the tasks of the project and organization.
4.3.3 Concepts Related to the Project's Defined Software Process

**Description of project's defined software process**

The description of the project's defined software process is the operational definition of the software process used by the project. The project's defined software process is a well-characterized and understood software process, described in terms of software standards, procedures, tools, and methods. It is developed by tailoring the organization's standard software process to fit the specific characteristics of the project.

This tailoring includes selecting a software life cycle from those approved by the organization and modifying the organization's standard software process to fit the specific characteristics of the project.

The project's defined software process provides the basis for planning, performing, and improving the activities of the managers and technical staff performing the project's tasks and activities. It is possible for a project to have more than one defined software process (e.g., for the operational software and for the test support software) or to have one defined software process for two or more similar projects.

**Stages**

A stage is a partition of the software effort that is of a manageable size and that represents a meaningful and measurable set of related tasks which are performed by the project. A stage is usually considered a subdivision of a software life cycle and is often ended with a formal review prior to the onset of the following stage.
A library of software process-related documentation is established to (1) store process documents that are potentially useful to other current and future projects, particularly as they relate to the organization's standard software process, and (2) make them available for sharing across the organization. This library contains example documents and document fragments, which are expected to be of use to future projects when they are tailoring the organization's standard software process. The examples may cover subjects such as a project's defined software process, standards, procedures, software development plans, measurement plans, and process training materials. This library is an important resource that can help to reduce the amount of effort required to start a new project, by providing examples of successful projects as a starting point.
Interpreting the CMM

Guidelines and criteria for tailoring

The organization’s standard software process is described at a general level that may not be directly usable by a project. Guidelines are established to guide the software projects in (1) selecting a software life cycle from those approved for use and (2) tailoring and elaborating the organization’s standard software process and the selected software life cycle to fit the specific characteristics of the project.

These guidelines and criteria help ensure that there is a common basis across all software projects for planning, implementing, measuring, analyzing, and improving the projects’ defined software processes.

Organization’s software process database

The organization’s software process database is a database established to collect and make available data on the software processes and resulting software work products, particularly as they relate to the organization’s standard software process. The database contains or references both the actual measurement data and the related information needed to understand the measurement data and assess it for reasonableness and applicability.

Examples of process and work product data include estimates of software size, effort, and cost; actual data on software size, effort, and cost; productivity data; peer review coverage and efficiency; and number and severity of defects found in the software code.
Interpreting the CMM

Software process element

A software process element is a constituent element of a software process description. Each process element covers a well-defined, bounded, closely related set of tasks (e.g., software estimating element, software design element, coding element, and peer review element). The descriptions of the process elements may be templates to be filled in, fragments to be completed, abstractions to be refined, or complete descriptions to be modified or used unmodified.

Description of software life cycles approved for use

A software life cycle is the period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept stage, requirements stage, design stage, implementation stage, test stage, installation and checkout stage, operation and maintenance stage, and sometimes, retirement stage [IEEE-STD-610].

Because an organization may be producing software for a variety of contractual and/or commercial customers and users, one software life cycle may not be appropriate for all situations. Therefore, the organization may identify more than one software life cycle for use by the projects. These software life cycles are typically obtained from software engineering literature and may be modified for the organization. These software life cycles are available to be used, in combination with the organization's standard software process, in developing a project's defined software process.
Interpreting the CMM

**Organization’s standard software process**

An organization’s standard software process is the operational definition of the basic process that guides the establishment of a common software process across the software projects in the organization. It describes the fundamental software process elements that each software project is expected to incorporate into its defined software process. It also describes the relationships (e.g., ordering and interfaces) between these software process elements. It guides the establishment of a common software process across the software development and maintenance projects in the organization.

The relationship between software process elements is sometimes referred to as a "software process architecture."

The organization’s standard software process forms the basis for the projects’ defined software processes. It provides continuity in the organization’s process activities and is the reference for the measurements and long-term improvement of the software processes used in the organization.

**Software process architecture**

The software process architecture is a high-level (i.e., summary) description of the organization’s standard software process. It describes the ordering, interfaces, interdependencies, and other relationships between the software process elements of the organization’s standard software process. It also describes the interfaces, dependencies, and other relationships to other external processes (e.g., system engineering, hardware engineering, and contract management).
4.3.2 Concepts Related to the Organization's Software Process Assets

The organization establishes and maintains a set of software process assets as shown in Figure 4.1. These software process assets include:

- the organization's standard software process (including the software process architecture and software process elements),
- the descriptions of software life cycles approved for use,
- the guidelines and criteria for tailoring the organization's standard software process,
- the organization's software process database, and
- the library of software process-related documentation.

The software process assets are available for use by the projects in developing, maintaining, and implementing their defined software process.

An organization may bundle the software process assets in many ways, depending on its approach to establishing its standard software process. For example, the description of the software life cycle may be an integral part of the organization's standard software process. Another example is that parts of the library of software process-related documentation may be stored in the organization's software process database.
Interpreting the CMM

Figure 4.1 Conceptual Software Process Framework Used in the CMM
4.3.1 Process Definition Concepts

A fundamental concept that supports the approach taken by the SEI in its process definition work is that processes can be developed and maintained in a way similar to the way products are developed and maintained. There must be:

- requirements that define what process is to be described,
- an architecture and design that provide information on how the process will be defined,
- implementation of the process design in a project or organizational situation,
- validation of the process description via measurement, and
- deployment of the process into widespread operation within the organization or project for which the process is intended.

Using the analogy of product development, a framework for software process development and maintenance has evolved that translates these concepts into ones which are more specific to the process development discipline (similar to the specificity of terminology used for developing real-time embedded systems versus management information systems). The key elements of this framework are illustrated in Figure 4.1 and described briefly below.

For further reading on the concepts of process definition that are being developed within the process engineering community, refer to the paper, "Software Process Development and Enactment: Concepts and Definitions" [Feiler 92].
4.3 Interpreting Software Process Definition

Software process definition is fundamental for achieving higher levels of maturity. This section discusses aspects of software process definition which are helpful in using the key practices related to process definition, beginning with Organization Process Definition at Level 3.

A fundamental concept of process definition in the CMM is the organization's standard software process. An organization's standard software process is the operational definition of the basic process that guides the establishment of a common software process across the software projects in the organization. It describes the fundamental software process elements that each software project is expected to incorporate into its defined software process. It also describes the relationships (e.g., ordering and interfaces) between these software process elements. It establishes a consistent way of performing the software activities across the organization and is essential for long-term stability and improvement.

At the organizational level, the organization's standard software process needs to be described, managed, controlled, and improved in a formal manner. At the project level, the emphasis is on the useability of the project's defined software process and the value it adds to the project. A project's defined software process is the operational definition of the software process used by the project. The project's defined software process is a well-characterized and understood software process, described in terms of software standards, procedures, tools, and methods. It is developed by tailoring the organization's standard software process to fit the specific characteristics of the project.

The key practices in Organization Process Definition are presented using terms that reflect an approach to process definition that supports both stability and flexibility. This approach is depicted in Figure 4.1, and its key elements are described in the following paragraphs.
Project management oversight on both a periodic and event-driven basis

The template "both on a periodic and event-driven basis" is used in these key practices to emphasize that projects have needs for different types of review at different stages and depending on the project characteristics. Project management should maintain an ongoing awareness of the status of the software effort and be informed when significant events on the software project occur. Examples include project management participation in formal reviews, such as critical design reviews, as well as reviews which encompass process issues such as status of process improvement planning and resolution of process non-compliance issues.

At the project level, project management oversight is expected to be at a more detailed level than that of senior management, reflecting project management’s more active involvement in the operational aspects of a project.

Software quality assurance activities

The particular activities that are considered appropriate for review and/or audit by the software quality assurance (SQA) group are described as a key practice. There are particular cases where SQA verification activities are not described, such as for the Training Program and Intergroup Coordination key process areas. These are key process areas that are at the boundary between the software project and the organization, where the SQA group would not be expected to have authority.
Interpreting the CMM

**Senior management oversight on a periodic basis**

The primary purpose of periodic reviews by senior management is to provide awareness of, and insight into, software process activities at an appropriate level of abstraction and in a timely manner. The time between reviews should meet the needs of the organization and may be lengthy, as long as adequate mechanisms for exception reporting are available.

The scope and content of senior management reviews will greatly depend on which senior manager is involved in the review. Reviews by the senior manager responsible for all software activities of an organization are expected to occur on a different schedule, and address different topics, from a review by the senior executive of the entire organization. Senior management reviews would also be expected to cover different topics, or similar topics at a higher level of abstraction, than project management oversight reviews.
4.2.4 Measurement and Analysis

The key practices in the Measurement and Analysis common feature describe basic measurement practices that are necessary to determine status related to the Activities Performed common feature of the key practices. Measurements that are inherently part of the activities of the key process area are contained under the Activities Performed common feature.

Examples of suggested measurements are expressed as supplementary information, because the variability in project environments may lead to different measurement needs and approaches.

4.2.5 Verifying Implementation

The Verifying Implementation common feature generally contains key practices that relate to oversight by senior management and project management, as well as specific verification activities that the software quality assurance group or others are expected to perform to verify that the key practices are being performed properly.
Interpreting the CMM

Placed under configuration management versus managed and controlled

Some software work products, e.g., the software design and the code, should have baselines established at predetermined points. These baselines are formally reviewed and agreed on and serve as the basis for further development. A rigorous change control process is applied to baselined items. These baselines provide control and stability when interacting with the customer. This is sometimes referred to as baseline configuration management. The phrase "placed under configuration management" is used for such software work products.

When control of the configuration is exercised by the developers, it is usually referred to as developmental configuration management. Some items under developmental configuration management may be placed under baseline configuration management at predetermined points in their development. The phrase "placed under configuration management" can be interpreted as extending to developmental configuration management, but a valid minimal interpretation is that only baseline configuration management is required.

Some software work products, such as estimates or the software development plan, which may not have to be under configuration management, still need to be "managed and controlled." This phrase is used to characterize the process of identifying and defining software work products that are not part of a baseline and, therefore, are not placed under configuration management but that must be controlled for the project to proceed in a disciplined manner. "Managed and controlled" implies that the version of the work product in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).
In Software Project Tracking and Oversight at Level 2, many of the key practices use the phrase, "... is tracked... corrective actions are taken as appropriate." In Integrated Software Management at Level 3, many of the similar key practices use the phrase, "is managed." This difference in wording reflects the project's lack of a completely defined software process at Level 2. Management actions are likely to be reactions to actual problems. At Level 3, the project has a complete defined software process, and the relationships between the various software work products, tasks, and activities are well defined. Management is better able to anticipate problems and proactively prevent them from occurring. When interventions are required, the effect on the entire software process is understood, and these interventions can be more effectively defined and applied.

At a review, a software work product, or set of work products, is presented to managers, the customer, end users, or other interested individuals for their comment or approval. Reviews typically occur at the end of a task. At a peer review, a software work product, or set of work products, is presented to the producer’s colleagues to identify defects. Managers, the customer, and end users are typically not present in a peer review. Peer reviews are an integral, in-process part of a task. They are performed so that defects can be removed early, leading to higher productivity and high-quality products. Some software work products will be reviewed; some will undergo peer review; and some will undergo both peer reviews and reviews.
Interpreting the CMM

Customer-supplier relationship

The customer may be internal or external to the organization. An example of an internal customer is a marketing group; an example of an external customer is the DoD. The user may also differ from the customer, as is typically the case in the DoD contracting environment. The CMM is expressed in terms of an external customer who is procuring a system with a critical software component.

Where necessary, the boundaries between groups, as stated in the CMM, must be appropriately interpreted. For example, in software-only procurements, there may be no system engineering group between the customer and the software engineering group. In such a case, customer requirements, system requirements, and allocated requirements may be synonymous, and the responsibilities of the system engineering group will be divided between the customer and the software engineering group.
The system requirements allocated to software, usually referred to as the "allocated requirements" in the CMM, are the subset of the system requirements that are to be implemented in the software components of the system. The allocated requirements are a primary input to the software development plan. Software requirements analysis elaborates and refines the allocated requirements and results in software requirements which are documented.

Customer requirements involve a complete system, not just software. In the CMM, discussion of customer requirements centers on those customer requirements to be implemented in software. The allocation of system requirements to hardware, software, etc., is typically done by a system engineering group as part of the overall system design. The system requirements allocated to the software project are usually referred to as the "allocated requirements" in the CMM and include both technical requirements (functionality, performance, etc.) and nontechnical requirements (delivery dates, cost, etc.).
Interpreting the CMM

According to a documented procedure, A documented procedure is usually needed so that the individuals responsible for a task or activity are able to perform it in a repeatable way and so that others with general knowledge of the area will be able to learn and perform the task or activity in the same way. This is one aspect of institutionalizing a process.

The formality and level of detail of a documented procedure can vary significantly, from a hand-written individual desk procedure to a formal organizational standard operating procedure. The formality and level of detail depends on who will perform the task or activity (e.g., individual or team), how often it is performed, the importance and intended use of the results, and the intended recipients of the results.
In cases where formal plans are called out, there are usually two key practices that specifically address the planning activities: a key practice that requires that the plan be developed or revised according to a documented procedure, and one that requires that the activities of the key process area be based on the plan.

The subpractices referring to a documented procedure generally cover what the inputs to the plan need to be, as well as the expected steps for obtaining commitment and support required for the plan. These subpractices identify the typical reviewers of the plan. They also highlight what levels of approval would be expected.

The subpractices that refer to the plan being the basis for activities describe the expected contents of the plan under discussion. Depending on the type of plan and need for organizational flexibility in covering the general topics of the plan, varying levels of detail are provided to describe the plan's contents.

Informal plans are usually described by a single key practice. The subpractices include information about the contents of the plan as well as the procedure for developing or revising the plan.
Interpreting the CMM

Types of plans

Two major types of plans are described in the key practices: formal plans (e.g., software development plan, software quality assurance plan, and software configuration management plan) and informal plans (e.g., peer review plan, risk management plan, and technology management plan).

The informal plans will typically be documented as part of a formal plan (e.g., the peer review plan may be documented as part of the software development plan) or as an adjunct to a formal plan (e.g., peer review schedules may be a section in the software development plan). Formal plans require a high degree of management commitment, both from the standpoint of creating them and ensuring that they are followed. In contractual environments, these plans are usually deliverable to the customer who contracted the effort.
Interpreting the CMM

**Orientation**

In some key process areas, key practices that describe orientation are found. The term orientation is used broadly to indicate less depth of skill or knowledge being transferred than would be expected via training. Orientation is an overview or introduction to a topic for those overseeing or interfacing with the individuals responsible for performing in the topic area.

**Prerequisite Items**

Some key process areas contain key practices that express a need for prerequisite items; for example, a software development plan is a prerequisite for Software Project Tracking and Oversight. In some cases, these are prerequisites that would be expected as outputs from the activities of another key process area. In other cases, they are items expected to be obtained from outside the realm of the software project (e.g., the system requirements allocated to software are a prerequisite for Requirements Management).

In keeping with the CMM philosophy of highlighting "key" practices, not all prerequisite items are listed for each key process area. Only those that have been found to be particularly critical for implementing the key process area are cited in the CMM.

**4.2.3 Activities Performed**

Of all the common features, Activities Performed shows the greatest amount of structural variability, because the implementation activities for the key process areas vary in level of detail, organizational focus (e.g., project or organization), and need for planning and documentation. Some generalizations are highlighted below.
Interpreting the CMM

**Training**

The CMM’s context for the term training is somewhat broader than might normally be considered when using the term. Training is provided to make an individual proficient with specialized instruction and practice. This training may include informal as well as formal vehicles for transferring skills and knowledge to the individuals in the organization. Although classroom training is a common mechanism that many organizations use to build the skills of their employees, the CMM also accommodates other vehicles, such as facilitated video, computer aided instruction, or formal mentoring and apprenticeship programs. The Training Program key process area describes the specific practices related to these training vehicles.

Two templates to describe training are generally found throughout the CMM. At Level 2, the phrase "receive training" is used. At Levels 3 and above, the phrase "receive required training" is used. The intention in using these different templates is to recognize that training at Level 2 is not likely to have been institutionalized across the organization. At Levels 3 and above, the key practices of the Training Program key process area are expected to govern the organization’s training activities.

In all the key process areas, potential training topics are expressed as example boxes, to recognize that different organizational situations are likely to drive different specific training needs.
4.2.2 Ability to Perform

Most key process areas contain a key practice that reflects the need for adequate resources and funding for the activities covered by the key process area. These resources and funding, described by the subpractices, generally fall into three categories: access to special skills, adequate funding, and access to tools. Tools that may be of use in performing the activities of the key process area are listed as examples.

The word "funding" is used, rather than "budget," to emphasize that what is delivered and used is more pertinent to the actual process than what was promised.
4.2 Interpreting the Common Features

Within each common feature of the key practices, certain phrases and conventions were used to provide continuity and consistency between the key process areas. The major structural conventions are described below, arranged by common feature.

4.2.1 Commitment to Perform

*Policy Statements*  
Where policy statements are used, they generally refer to the project following a written, organizational policy for the practices of that key process area. This is to emphasize the connection between organizational commitment and the projects that are actually performing the work.

The subpractices for the policy statement generally summarize activities that are covered later in the key process area and are particularly suitable to institutionalization via a written policy.

In some key process areas (e.g., Organization Process Focus), the focus of the activities for the key process area is the organization, not the project. In those cases, the policy statement is reworded and refers to the organization following a written policy.

*Leadership*  
In some key process areas, Commitment to Perform contains a statement that addresses the assignment of a leadership role (e.g., project software manager) or that describes particular sponsorship activities, which are necessary for the key process area to be successfully institutionalized.
4 Interpreting the CMM

4.1 Interpreting the Key Practices

The intention in setting down the key practices is not to require or espouse a specific model of the software life cycle, a specific organizational structure, a specific separation of responsibilities, or a specific management and technical approach to development. The intention, rather, is to provide a description of the essential elements of an effective software process.

The key practices are intended to communicate principles that apply to a wide variety of projects and organizations, that are valid across a range of typical software applications, and that will remain valid over time. Therefore, the approach is to describe the principles and leave their implementation up to each organization, according to its culture and the experiences of its managers and technical staff.

Although the key practices are meant to be independent of any particular implementation, specific terms and examples are consistently used in stating the key practices to improve clarity. This section describes the conventions used in the CMM for roles, responsibilities, relationships, products, and activities. Organizations using the key practices should be aware of these conventions and map them appropriately to their own organization, project, and business environment.

The glossary in Appendix B contains definitions of terms, including those described in this section and others.
Using the Key Practice Pages
Using the Key Practice Pages

Figure 3.1 Example of Key Practice Statements

Ability 3

Key practice

Adequate resources and funding are provided for planning the software project.

1. Where feasible, experienced individuals, who have expertise in the application domain of the software project being planned, are available to develop the software development plan.

2. Tools to support the software project planning activities are made available.

Subpractice

Supplementary information

Examples of support tools include:
- spreadsheet programs,
- estimating models, and
- project planning/scheduling programs.

Activity 1

Common feature

Activities performed

The software engineering group participates on the project proposal team.

1. The software engineering group is involved in:
   - proposal preparation and submission,
   - clarification discussions and submissions, and
   - negotiations of changes to commitments that affect the software project.

2. The software engineering group reviews the project’s proposed commitments.

The software managers, software engineers, and other individuals involved in the software project planning are trained in the software estimating and planning procedures applicable to their areas of responsibility.
Using the Key Practice Pages

**Supplementary information**

Supplementary information is boxed following the key practices. The supplementary information includes examples, elaborations, and references to other key process areas.

When the subpractices or the supplementary information underneath a key practice extends to another page, the number of the key practice is shown in parentheses at the start of the new page to indicate that the information on that page is a continuation of the key practice on the previous page.
3 Using the Key Practice Pages

The key practices are grouped by maturity level, and each maturity level is separated by a tab page. The tab page includes a description of the maturity level, a list of the key process areas for that maturity level, and the page number where each key process area begins.

Each key process area contains:

- a brief description of the key process area,
- the goals for the key process area, and
- the key practices.

The key practices themselves are grouped into the five common features (Commitment to Perform, Ability to Perform, Activities Performed, Measurement and Analysis, and Verifying Implementation) and are presented in a hierarchical format, as shown in Figure 3.1, an example page from the key practices. The key practices include:

**Key practices**

The key practices, also known as top-level key practices, state the fundamental policies, procedures, and activities for the key process area. They are identified in bold and are numbered within each common feature. For example, the first key practice in the common feature of Activities Performed is identified as Activity 1.

**Subpractices**

Subpractices, also known as subordinate key practices, are listed beneath the top-level key practices and describe what one would expect to find implemented for the top-level key practice. The subpractices can be used to help determine whether or not the key practices are implemented satisfactorily.
Overview of the Capability Maturity Model
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. The Activities Performed by projects or the organization provide the largest category of key practices because they describe the actual implementation of the key process area. Key practices under the other common features are equally important, however, for they address what must be done to support and institutionalize the key process area.
Overview of the Capability Maturity Model

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.
Overview of the Capability Maturity Model

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.

2.7 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. In adapting the key practices of a key process area to a specific project situation, the goals can be used to determine whether the adaptation is a reasonable rendering of the practices. Similarly, when assessing or evaluating alternative ways to implement a key process area, the goals can be used to determine if the alternatives satisfy the intent of the key process area. Please refer to "Capability Maturity Model for Software, Version 1.1" [Paulk93a] and Section 4.5, Applying Professional Judgment, of this document for more information on interpreting the goals in an organization.

2.8 Common Features

The key practices in each key process area are organized by a set of 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 common features also group and order the key practices in a sequence helpful for organizations using them. The five common features are listed below:
Overview of the Capability Maturity Model

The key process areas are categorized in Figure 2.4 into three broad categories: Management, Organizational, and Engineering processes. The Management process category contains the project management activities as they evolve from planning and tracking at Level 2, to managing according to a defined software process at Level 3, to quantitative management at Level 4, to innovative management in a constantly changing environment at Level 5. The Organizational process category contains the cross-project responsibilities as the organization matures, beginning with a focus on process issues at Level 3, continuing to a quantitative understanding of the process at Level 4, and culminating with the management of change in an environment of continuous process improvement at Level 5. The Engineering process category contains the technical activities, such as requirements analysis, design, code, and test, which are performed at all levels, but that evolve toward an engineering discipline at Level 3, statistical process control at Level 4, and continuous measured improvement at Level 5.

Note that at Levels 4 and 5 there are key process areas that span these process categories. This helps identify potential new key process areas for CMM v2 as Levels 4 and 5 become better understood.

2.6 The 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.
<table>
<thead>
<tr>
<th>Processes Categories</th>
<th>Management</th>
<th>Organizational</th>
<th>Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Levels</td>
<td>Software project planning, management, etc.</td>
<td>Senior management review, etc.</td>
<td>Requirements analysis, design, code, test, etc.</td>
</tr>
<tr>
<td>5 Optimizing</td>
<td>Technology Change Management</td>
<td>Process Change Management</td>
<td>Defect Prevention</td>
</tr>
<tr>
<td>4 Managed</td>
<td>Quantitative Process Management</td>
<td></td>
<td>Software Quality Management</td>
</tr>
<tr>
<td>3 Defined</td>
<td>Integrated Software Management</td>
<td>Organization Process Focus</td>
<td>Software Product Engineering</td>
</tr>
<tr>
<td></td>
<td>Intergroup Coordination</td>
<td>Organization Process Definition</td>
<td>Peer Reviews</td>
</tr>
<tr>
<td>2 Repeatable</td>
<td>Requirements Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Project Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Project Tracking &amp; Oversight</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Subcontract Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Quality Assurance</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Configuration Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Initial</td>
<td>Ad Hoc Processes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.4 The Key Process Areas Assigned to Process Categories
Overview of the Capability Maturity Model

- 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, as is described in Integrated Software Management. Process changes of general value are transitioned to other software projects, as is described in Process Change Management.

- 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, as is described in Process Change Management. 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. Process Change Management takes the incremental improvements of Defect Prevention and the innovative improvements of Technology Change Management and makes them available to the entire organization.

By definition, key process areas are expressed at a single maturity level. There are, however, relationships between the key process areas, and improvements in a specific management or technical area need not be restricted to a single key process area. Figure 2.4 illustrates these relationships. Organizations may work on higher level key process areas before they have achieved lower level maturity levels, and attention must continue to be focused on lower level key process areas even when key process areas at higher maturity levels have been achieved.
Overview of the Capability Maturity Model

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 is called out in Software Product Engineering and that can be implemented via Fagan-style inspections [Fagan86], structured walkthroughs, or a number of other collegial review methods [Freedman90].

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 two key process areas at this level, Quantitative Process Management and Software Quality Management, are highly interdependent, as is described below:

- 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. Quantitative Process Management adds a comprehensive measurement program to the practices of Organization Process Definition, Integrated Software Management, Intergroup Coordination, and Peer Reviews.

- 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. Software Quality Management applies a comprehensive measurement program to the software work products described in Software Product Engineering.

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. Descriptions of each of the key process areas for Level 5 are given below:
Overview of the Capability Maturity Model

- 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, which are described in Organization Process Definition. This tailoring is based on the business environment and technical needs of the project, as described in Software Product Engineering. Integrated Software Management evolves from Software Project Planning and Software Project Tracking and Oversight at Level 2.

- 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. Intergroup Coordination is the interdisciplinary aspect of Integrated Software Management that extends beyond software engineering; not only should the software process be integrated, but the software engineering group's interactions with other groups must be coordinated and controlled.

- The purpose of Peer Reviews is to remove defects from the software work products early and efficiently. An important corollary effect is to
Overview of the Capability Maturity Model

- 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. Software Quality Assurance is an integral part of most software engineering and management processes.

- 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. Software Configuration Management is an integral part of most software engineering and management processes.

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. Descriptions of each of the key process areas for Level 3 are given below:

- 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 primary result of the Organization Process Focus activities is a set of software process assets, which are described in Organization Process Definition. These assets are used by the software projects, as is described in Integrated Software Management.

- 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 cumulative, long-term benefits to the organization. These assets provide a stable foundation that can be institutionalized via mechanisms such as training, which is described in Training Program.
Overview of the Capability Maturity Model

The key process areas at Level 2 focus on the software project's concerns related to establishing basic project management controls. Descriptions of each of the key process areas for Level 2 are given below:

- 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 (as described in Software Project Planning) and managing (as described in Software Project Tracking and Oversight) the software project. Control of the relationship with the customer depends on following an effective change control process (as described in Software Configuration Management).

- 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 (as described in Software Project Tracking and Oversight). Without realistic plans, effective project management cannot be implemented.

- 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. It combines the concerns of Requirements Management, Software Project Planning, and Software Project Tracking and Oversight for basic management control, along with necessary coordination of Software Quality Assurance and Software Configuration Management, and applies this control to the subcontractor as appropriate.
Figure 2.3 The Key Process Areas by Maturity Level
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.

2.5 The Key Process Areas of the CMM

Figure 2.3 lists the key process areas for each maturity level in the CMM.
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. The key process areas are building blocks
that indicate the areas an organization should focus on to improve its
software process. Key process areas identify the issues that must be
addressed to achieve a maturity level.
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 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. When these limits are exceeded, action is taken to correct the situation. Software products are of predictably high quality.

2.4.5 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.

The software process capability of Level 5 organizations can be characterized as continuously improving because Level 5 organizations are continuously
Overview of the Capability Maturity Model

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.

2.4.4 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 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, 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 strong customer-supplier relationship.

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.

2.4.3 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.
Overview of the Capability Maturity Model

Software engineering practices are undermined by ineffective planning and reaction-driven commitment systems.

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.

The software process capability of Level 1 organizations is unpredictable because the software process is constantly changed or modified as the work progresses (i.e., the process is ad hoc). Schedules, budgets, functionality, and product quality are generally unpredictable. Performance depends on the capabilities of individuals and varies with their innate skills, knowledge, and motivations. There are few stable software processes in evidence, and performance can be predicted only by individual rather than organizational capability.

2.4.2 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. An objective in achieving Level 2 is to institutionalize effective management processes for software projects, which allow organizations to repeat successful practices developed on earlier projects, although the specific processes implemented by the projects may differ. An effective process can be characterized as practiced, documented, enforced, trained, measured, and able to improve.
2.4.1 Level 1 - The Initial Level

At the Initial Level, the organization typically does not provide a stable environment for developing and maintaining software. When an organization lacks sound management practices, the benefits of good
Overview of the Capability Maturity Model

Key practices

Each key process area is described in terms of key practices that, when implemented, help to satisfy the goals of that key process area. The key practices describe the infrastructure and activities that contribute most to the effective implementation and institutionalization of the key process area.

For example, one of the practices from the Software Project Planning key process area is "The project's software development plan is developed according to a documented procedure."

2.4 Definition of the CMM Maturity Levels

As organizations establish and improve the software processes by which they develop and maintain their software work products, they progress through levels of maturity. Figure 2.2 shows the five maturity levels of the CMM.

Each maturity level provides a layer in the foundation for continuous process improvement. Each key process area comprises a set of goals that, when satisfied, stabilize an important component of the software process. Achieving each level of the maturity model institutionalizes a different component in the software process, resulting in an overall increase in the process capability of the organization.
Overview of the Capability Maturity Model

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.

An example of a goal from the Software Project Planning key process area is "Software estimates are documented for use in planning and tracking the software project." See "Capability Maturity Model for Software, Version 1.1" [Paulk93a] and Section 4.5, Applying Professional Judgment, of this document for more information on interpreting the goals.

Common Features
The key practices are divided among five Common Features sections: Commitment to Perform, Ability to Perform, Activities Performed, Measurement and Analysis, and Verifying Implementation. The common features are attributes that indicate whether the implementation and institutionalization of a key process area is effective, repeatable, and lasting.

The Activities Performed common feature describes implementation activities. The other four common features describe the institutionalization factors, which make a process part of the organizational culture.
Overview of the Capability Maturity Model

The components of the CMM include:

* **Maturity levels**
  A maturity level is a well-defined evolutionary plateau toward achieving a mature software process. The five maturity levels provide the top-level structure of the CMM.

* **Process capability**
  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.

* **Key process areas**
  Each maturity level is composed of key process areas. Each key process area identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for establishing process capability at that maturity level. The key process areas have been defined to reside at a single maturity level. For example, one of the key process areas for Level 2 is Software Project Planning.
Figure 2.1 The Structure of the Capability Maturity Model
Overview of the Capability Maturity Model

- reflects the best of the state of the practice;
- reflects the needs of individuals performing software process improvement, software process assessments, or software capability evaluations;
- is documented; and
- is publicly available.

Additional knowledge and insight into software process maturity has been gained since the earlier versions of the maturity model. This insight has been gained by:

- studying non-software organizations,
- performing and observing software process assessments and software capability evaluations,
- soliciting and analyzing change requests to the model,
- participating in meetings and workshops with industry and government representatives, and
- soliciting feedback from industry and government reviewers.

Using this additional knowledge, the Capability Maturity Model and its practices have been revised, creating CMM v1.1.

2.3 Structure of the CMM

The CMM is composed of five maturity levels. With the exception of Level 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. This structure of the CMM is illustrated in Figure 2.1.
2 Overview of the Capability Maturity Model

2.1 Introducing the Capability Maturity Model

The Capability Maturity Model for Software (CMM) is a framework that describes the key elements of an effective software process. The CMM describes an evolutionary improvement path from an ad hoc, immature process to a mature, disciplined process.

The CMM covers practices for planning, engineering, and managing software development and maintenance. When followed, these key practices improve the ability of organizations to meet goals for cost, schedule, functionality, and product quality.

The CMM establishes a yardstick against which it is possible to judge, in a repeatable way, the maturity of an organization's software process and compare it to the state of the practice of the industry [Kitson92]. The CMM can also be used by an organization to plan improvements to its software process.

2.2 Sources of the CMM

The Software Engineering Institute (SEI) developed an initial version of a maturity model and maturity questionnaire at the request of the government and with the assistance of the MITRE Corporation. Throughout the development of the model and the questionnaire, the SEI has paid attention to advice from practitioners who are involved in developing and improving software processes. Our objective has been to provide a model that:

- is based on actual practices;
Introducing the Key Practices Document
Following the overview, the key practices for the key process areas of the CMM are described. For those who want to get a quick sense of the key practices, without the rigor that is needed in applying them, an abridgment of the key practices is provided in Appendix C.

In the appendices are a list of the references cited in this document, a glossary of terms used in this document, an abridgment of the key practices, the change history for this document, and an index of terms contained in this document.

1.4 Expected Use of this Document

If you are not familiar with the CMM, you should first read the paper, "Capability Maturity Model for Software, Version 1.1" [Paulk93a] and all four chapters in this overview before trying to use the key practices.

If you are already familiar with the CMM and how it is structured, you may want to go directly to the fourth chapter for advice on how to interpret the key practices.
Introducing the Key Practices Document

- "Capability Maturity Model for Software, Version 1.1" [Paulk93a], and
- this document, "Key Practices of the Capability Maturity Model, Version 1.1" [Paulk93b].

"Capability Maturity Model for Software, Version 1.1" contains an introduction to the model, descriptions of the five maturity levels, an operational definition of the CMM and its structure, a discussion of how organizations can use the maturity model, and some remarks on the future directions of the CMM.

"Key Practices of the Capability Maturity Model, Version 1.1," contains the key practices that correspond to the key process areas at each maturity level of the CMM and information to help interpret the key practices.

The maturity questionnaire and other process products are derived from the key practices of the Capability Maturity Model. Other SEI process products that support software process improvement, software process assessment, and software capability evaluation include training courses, handbooks, and site visit guides.

1.3 Organization of this Document

This first chapter gives an overview of the CMM and of this document. In the next three chapters of the overview are:

- an overview of the CMM and its constituent parts,
- a description of how to use the format of the key practices, and
- a description of ways to use and interpret the key practices.
Introducing the Key Practices Document

- by anyone wanting to identify the key practices that are needed to achieve the next maturity level in the CMM,
- by organizations wanting to understand and improve their capability to develop software effectively,
- by acquisition organizations or prime contractors wanting to identify the risks of having a particular organization perform the work of a contract,
- by the SEI as the basis for developing process products, such as the maturity questionnaire, and
- by instructors preparing teams to perform software process assessments or software capability evaluations.

1.2 Relationship of this Document to Other Documents

The two documents that provided the initial foundation for the CMM are:
- "Characterizing the Software Process" [Humphrey88], and
- Managing the Software Process [Humphrey89].

Version 1.0 of the CMM was released in August of 1991 in two technical reports:
- "Capability Maturity Model for Software" [Paulk91], and
- "Key Practices of the Capability Maturity Model" [Weber91].

This initial release of the CMM was revised during 1992. To understand and use the current version of the CMM, two documents are needed:
Introducing the Key Practices Document

- software process assessments, in which a trained team of software professionals determines the state of an organization's current software process, determines the high-priority software process-related issues facing an organization, and obtains the organizational support for software process improvement; and

- software capability evaluations, in which a trained team of professionals identifies contractors who are qualified to perform the software work or monitors the state of the software process used on an existing software effort.

This document describes the key practices that correspond to each maturity level in the CMM. It is an elaboration of what is meant by maturity at each level of the CMM and a guide that can be used for software process improvement, software process assessments, and software capability evaluations.

The key practices of the CMM are expressed in terms of what is expected to be the normal practices of organizations that work on large, government contracts. In any context in which the CMM is applied, a reasonable interpretation of how the practices would be applied should be used. Guidelines on interpreting the CMM are contained in Chapter 4 of this document. The CMM must be appropriately interpreted when the business environment of the organization differs significantly from that of a large contracting organization. The role of professional judgment in making informed use of the CMM must be recognized.

This document can be used in several ways:

- by anyone wanting to understand the key practices that are part of effective processes for developing or maintaining software,
1 Introducing the Key Practices Document

1.1 To the Reader

Developing reliable and usable software that is delivered on time and within budget is a difficult endeavor for many organizations. Products that are late, over budget, or that don’t work as expected also cause problems for the organization’s customers. As software projects continue to increase in size and importance, these problems become magnified. These problems can be overcome through a focused and sustained effort at building a process infrastructure of effective software engineering and management practices.

To build this process infrastructure, organizations producing software need ways to appraise their ability to perform their software process successfully. They also need guidance to improve their process capability. Customers, such as the Department of Defense (DoD), need ways to evaluate more effectively an organization’s capability to perform successfully on software engineering contracts. Prime contractors need ways to evaluate the capability of potential subcontractors.

To help organizations and customers like the DoD and prime contractors, the Software Engineering Institute (SEI) has developed the Capability Maturity Model for Software (CMM), that delineates the characteristics of a mature, capable software process. The progression from an immature, unrepeatable software process to a mature, well-managed software process also is described in terms of maturity levels in the model.

The CMM can be used for:

- software process improvement, in which an organization plans, develops, and implements changes to its software process;
Acknowledgments

they helped focus our effort and worked with us to evaluate and plan our actions to address the many comments received from industry and government reviewers. The current members are Constance Ahara, Kelley Butler, Bill Curtis, Conrad Czaplicki, Raymond Dion, Judah Mogilensky, Martin Owens, Mark Paulk, Sue Stetak, Charlie Weber, and Ronald Willis. Former members who worked with us on CMM v1.0 include Harry Carl, Jim Hess, Jerry Pixton, and Jim Withey.

Lastly, we would like to thank the many individuals who helped produce this document. We would like to thank the members of the Joint Program Office who expedited the approval process. We would also like to thank Ginny Redish and Renee Dutkowski from the American Institutes for Research for their help with editing and designing the document. We very much appreciate the efforts of Carolyn Tady, David White, and Debbie Punjack for their administrative support, and also Mary Beth Chrissis, Suzanne Couturiaux, and Mike Konrad who did the editing, formatting, and whatever else was necessary to get the document produced.
Acknowledgments

The SEI would like to thank the many people who were involved in the development of this document. The SEI has been developing and revising the Capability Maturity Model for Software (CMM) since 1988 based on the knowledge acquired from software process assessments, software capability evaluations, and feedback from both industry and government. This document was produced as part of the revision of CMM Version 1.0, which was released in August, 1991.

The conceptual framework for the capability maturity model was developed by Watts Humphrey. He has advised us as we refined the model and evolved the practices.

This document has taken many drafts to evolve into the current product. We would like to thank the individuals who have worked many hours developing ideas and formulating the practices for this model. In particular, we acknowledge the contributions of Charlie Weber, Mark Paulk, Cynthia Wise, Jim Withey, Mary Beth Chrissis, Suzanne Garcia, and Marilyn Bush.

Many other individuals have helped with their comments, criticisms, and suggestions. They include Ed Averill, Judy Bamberger, Anita Carleton, Bill Curtis, Susan Dart, Lionel Deimel, Peter Feiler, Julia Gale, Jim Hart, Ron Higuera, Purvis Jackson, Tim Kasse, David Kitson, Mike Konrad, Peter Malpass, Mark Manduke, Judah Mogilensky, Warren Moseley, Jim Over, George Pandelios, Bob Park, Jeff Perdue, Dick Phillips, Mike Rissman, Jim Rozum, and Christer von Schantz.

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. We would like to thank especially those who took time to participate in the March 1990 and April 1992 CMM Workshops, as well as those who took time to meet with us on other occasions to provide comments and recommendations. This effort could not have been accomplished without these practitioners lending their expertise to refine the model.

We would also like to thank the members of the CMM Advisory Board who helped guide us in our efforts. In addition to providing technical insights,
List of Figures

Figure 2.1 The Structure of the Capability Maturity Model .................. O-9
Figure 2.2 The Five Levels of Software Process Maturity .................. O-13
Figure 2.3 The Key Process Areas by Maturity Level .......................... O-19
Figure 2.4 The Key Process Areas Assigned to Process Categories ......... O-25
Figure 3.1 Example of Key Practice Statements ............................. O-33
Figure 4.1 Conceptual Software Process Framework Used in the CMM ................................................................................ O-52
Appendices

Appendix A: References ................................................................. A-1
Appendix B: Glossary of Terms .................................................. A-3
Appendix C: Abridged Version of the Key Practices .................. A-27
Appendix D: Change History ...................................................... A-67
Appendix E: Index ...................................................................... A-73
Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.4 Relationship Between the Project’s Defined Software Process and the Software Development Plan...</td>
<td>O-61</td>
</tr>
<tr>
<td>4.3.5 Life Cycles and the CMM</td>
<td>O-61</td>
</tr>
<tr>
<td>4.3.6 Technology and the CMM</td>
<td>O-62</td>
</tr>
<tr>
<td>4.3.7 Documentation and the CMM</td>
<td>O-62</td>
</tr>
<tr>
<td>4.3.8 Collection and Analysis of Process Data</td>
<td>O-63</td>
</tr>
<tr>
<td>4.4 Organizational Structure and Roles</td>
<td>O-64</td>
</tr>
<tr>
<td>4.4.1 Organizational Roles</td>
<td>O-64</td>
</tr>
<tr>
<td>4.4.2 Organizational Structure</td>
<td>O-69</td>
</tr>
<tr>
<td>4.4.3 Independence and Organizational Structure</td>
<td>O-73</td>
</tr>
<tr>
<td>4.5 Applying Professional Judgment</td>
<td>O-74</td>
</tr>
</tbody>
</table>

**Level 2 Key Practices**

<table>
<thead>
<tr>
<th>Practice</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Management</td>
<td>L2-1</td>
</tr>
<tr>
<td>Software Project Planning</td>
<td>L2-11</td>
</tr>
<tr>
<td>Software Project Tracking and Oversight</td>
<td>L2-29</td>
</tr>
<tr>
<td>Software Subcontract Management</td>
<td>L2-43</td>
</tr>
<tr>
<td>Software Quality Assurance</td>
<td>L2-59</td>
</tr>
<tr>
<td>Software Configuration Management</td>
<td>L2-71</td>
</tr>
</tbody>
</table>

**Level 3 Key Practices**

<table>
<thead>
<tr>
<th>Practice</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization Process Focus</td>
<td>L3-1</td>
</tr>
<tr>
<td>Organization Process Definition</td>
<td>L3-11</td>
</tr>
<tr>
<td>Training Program</td>
<td>L3-25</td>
</tr>
<tr>
<td>Integrated Software Management</td>
<td>L3-37</td>
</tr>
<tr>
<td>Software Product Engineering</td>
<td>L3-59</td>
</tr>
<tr>
<td>Intergroup Coordination</td>
<td>L3-83</td>
</tr>
<tr>
<td>Peer Reviews</td>
<td>L3-93</td>
</tr>
</tbody>
</table>

**Level 4 Key Practices**

<table>
<thead>
<tr>
<th>Practice</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantitative Process Management</td>
<td>L4-1</td>
</tr>
<tr>
<td>Software Quality Management</td>
<td>L4-19</td>
</tr>
</tbody>
</table>

**Level 5 Key Practices**

<table>
<thead>
<tr>
<th>Practice</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defect Prevention</td>
<td>L5-1</td>
</tr>
<tr>
<td>Technology Change Management</td>
<td>L5-17</td>
</tr>
<tr>
<td>Process Change Management</td>
<td>L5-31</td>
</tr>
</tbody>
</table>
Table of Contents

Acknowledgments........................................................................................................... O-vii

Overview
  1 Introducing the Key Practices Document............................................................... O-1
    1.1 To the Reader................................................................................................. O-1
    1.2 Relationship of this Document to Other Documents....................................... O-3
    1.3 Organization of this Document....................................................................... O-4
    1.4 Expected Use of this Document..................................................................... O-5
  2 Overview of the Capability Maturity Model.......................................................... O-7
    2.1 Introducing the Capability Maturity Model.................................................... O-7
    2.2 Sources of the CMM.................................................................................... O-7
    2.3 Structure of the CMM.................................................................................. O-8
      2.4.1 Level 1 - The Initial Level....................................................................... O-13
      2.4.2 Level 2 - The Repeatable Level............................................................... O-14
      2.4.3 Level 3 - The Defined Level..................................................................... O-15
      2.4.4 Level 4 - The Managed Level................................................................ O-16
      2.4.5 Level 5 - The Optimizing Level............................................................... O-17
    2.5 The Key Process Areas of the CMM................................................................. O-18
    2.6 The Key Practices.......................................................................................... O-26
    2.7 Goals............................................................................................................. O-27
    2.8 Common Features.......................................................................................... O-27
  3 Using the Key Practice Pages.................................................................................. O-31
  4 Interpreting the CMM............................................................................................ O-35
    4.1 Interpreting the Key Practices........................................................................ O-35
    4.2 Interpreting the Common Features.................................................................. O-35
      4.2.1 Commitment to Perform.......................................................................... O-36
      4.2.2 Ability to Perform................................................................................... O-37
      4.2.3 Activities Performed................................................................................. O-39
      4.2.4 Measurement and Analysis..................................................................... O-47
      4.2.5 Verifying Implementation........................................................................ O-47
    4.3 Interpreting Software Process Definition....................................................... O-50
      4.3.1 Process Definition Concepts..................................................................... O-51
      4.3.2 Concepts Related to the Organization’s Software Process Assets............. O-53
      4.3.3 Concepts Related to the Project’s Defined Software Process.................. O-58