AD-A263 403

Capability Maturity Model for Software, Version 1.1

Mark C. Paulk
Bill Curtis
Mary Beth Chrissis
Charles V. Weber

DTIC
ELECTE
APR 26 1993
S B D

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

93-08746

93 4 052

February 1993
CMU/SEI-93-TR-24
ESC-TR-93-177

Capability Maturity Model for Software, Version 1.1

Mark C. Paulk
Bill Curtis
Mary Beth Chrissis
Charles V. Weber

Approved for public release
Distribution unlimited

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

SEI Joint Program Office
ESC/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

The Software Engineering Institute is sponsored by the U.S. Department of Defense.
This report was funded by the U.S. Department of Defense.
Copyright © 1993 by Carnegie Mellon University.

This document is available through the Defense Technical Information Center. DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145.

Copies of this document are also available through the National Technical Information Service. For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161.

Copies of this document are also available from Research Access, Inc., 3400 Forbes Avenue, Suite 302, Pittsburgh, PA 15213.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Carnegie Mellon University
Software Engineering Institute

Permission to reproduce, in whole or in part, the volume of materials released by the Software Engineering Institute under the title *Capability Maturity Model for Software* is granted under the following conditions:

1. This letter must be reproduced with each copy.

2. All copies must include the copyright notice.

3. The materials are not to be used for commercial gain.

4. The materials are not to be distributed beyond your organization. Refer such requests to the SEI.

5. The materials are to be used in a manner consistent with the framework and methodology advanced by the SEI.

6. Carnegie Mellon University and the Software Engineering Institute are not to be construed as responsible for the results of analyses conducted as a result of this permission. In other words, neither CMU nor the SEI is to be held liable for your use of this material.

Sincerely,

Purvis M. Jackson

Purvis M. Jackson
Sr. Editor and Director
SEI Information Management

DTIC QUALITY INSPECTED 4

Accession For
NTIS CRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890
(412) 268-7700

Table of Contents

Acknowledgments v
To the Reader vii
What is the Purpose of This Paper? viii
Who Should Read This Paper? viii
How is This Paper Organized? ix
What Are the Other CMM Products? x
How Do You Receive More Information? xi
1 The Process Maturity Framework 1
1.1 Immature Versus Mature Software Organizations 1
1.2 Fundamental Concepts Underlying Process Maturity 3
1.3 Overview of the Capability Maturity Model 4
2 The Five Levels of Software Process Maturity 7
2.1 Behavioral Characterization of the Maturity Levels 9
2.1.1 Level 1 - The Initial Level 10
2.1.2 Level 2 - The Repeatable Level 10
2.1.3 Level 3 - The Defined Level 11
2.1.4 Level 4 - The Managed Level 12
2.1.5 Level 5 - The Optimizing Level 13
2.2 Understanding the Maturity Levels 14
2.2.1 Understanding the Initial Level 15
2.2.2 Understanding the Repeatable and Defined Levels 15
2.2.3 Understanding the Managed and Optimizing Levels 16
2.3 Visibility Into the Software Process 19
2.4 Process Capability and the Prediction of Performance 22
2.5 Skipping Maturity Levels 25
3 Operational Definition of the Capability Maturity Model 27
3.1 Internal Structure of the Maturity Levels 27
3.2 Maturity Levels 30
3.3 Key Process Areas 30
3.4 Common Features 37
3.5 Key Practices 39
4 Using the CMM 43
4.1 Software Process Assessment and Software Capability Evaluation Methods 44
4.2 Differences Between Software Process Assessments and Software Capability Evaluations 47
4.3 Other Uses of the CMM in Process Improvement 49

5 Future Directions of the CMM....51
5.1 What the CMM Does Not Cover....51
5.2 Near-Term Activities....51
5.3 Long-Term Activities....52
5.4 Conclusion....53
6 References....55
Appendix A: Goals for Each Key Process Area....59
A.1 The Key Process Areas for Level 2: Repeatable....59
A.2 The Key Process Areas for Level 3: Defined....61
A.3 The Key Process Areas for Level 4: Managed....62
A.4 The Key Process Areas for Level 5: Optimizing....63

List of Figures

Figure 2.1 The Five Levels of Software Process Maturity 8
Figure 2.2 The Juran Trilogy Diagram: Quality Planning, Quality Control, and Quality Improvement 17
Figure 2.3 A Management View of Visibility into the Software Process at Each Maturity Level 20
Figure 2.4 Process Capability as Indicated by Maturity Level 23
Figure 3.1 The CMM Structure 29
Figure 3.2 The Key Process Areas by Maturity Level 31
Figure 3.3 Building the CMM Structure: An Example of a Key Practice 40
Figure 4.1 Common Steps in Software Process Assessments and Software Capability Evaluations 45

Acknowledgments

The description of the Capability Maturity Model for Software was initially produced by a dedicated group of people who spent many hours discussing the model and its features and then trying to document it in CMM v1.0. This group consisted of Mark Paulk, Bill Curtis, Mary Beth Chrissis, Edward Averill, Judy Bamberger, Tim Kasse, Mike Konrad, Jeff Perdue, Charlie Weber, and Jim Withey.

This paper is based on the vision of Watts Humphrey, first director of the SEI's Software Process Program. It took several drafts to evolve this paper into the final product. Jim Withey, Mark Paulk, and Cynthia Wise produced an early draft in 1990. Watts Humphrey provided a second draft of the document, and Mark Paulk then took over the paper and remained book boss until the end. Mary Beth Chrissis and Bill Curtis helped Mark produce the CMM v1.0 revision of this paper in August, 1991. Mark Paulk produced the CMM v1.1 revision of the paper, which is this technical report.

At various stages, several people contributed to the concepts expressed in this paper. They include Joe Besselman, Marilyn Bush, Anita Carleton, Marty Carlson, Betty Deimel, Suzie Garcia, Richard Kauffold, Steve Masters, Mary Merrill, Jim Over, George Pandelios, Jane Siegel and Charlie Weber.

We appreciate the administrative help from Todd Bowman, Dorothy Josephson, Debbie Punjack, Carolyn Tady, Marcia Theoret, Andy Tsounos, and David White; and the editorial assistance from Mary Beth Chrissis, Suzanne Couturiaux, and Bill Pollak. Renne Dutkowski from the American Institutes for Research provided suggestions for the design of the document.

To the Reader

In November 1986, the Software Engineering Institute (SEI), with assistance from the Mitre Corporation, began developing a process maturity framework that would help organizations improve their software process. This effort was initiated in response to a request to provide the federal government with a method for assessing the capability of its software contractors. In September 1987, the SEI released a brief description of the process maturity framework [Humphrey 87a] and a maturity questionnaire [Humphrey87b]. The SEI intended the maturity questionnaire to provide a simple tool for identifying areas where an organization's software process needed improvement. Unfortunately, the maturity questionnaire was too often regarded as "the model" rather than as a vehicle for exploring process maturity issues.

After four years of experience with the software process maturity framework and the preliminary version of the maturity questionnaire, the SEI evolved the software process maturity framework into the Capability Maturity Model for Software (CMM) [Paulk91, Weber91]. The CMM is based on knowledge acquired from software process assessments and extensive feedback from both industry and government. By elaborating the maturity framework, a model has emerged that provides organizations with more effective guidance for establishing process improvement programs.

The initial release of the CMM, Version 1.0, was reviewed and used by the software community during 1991 and 1992. A workshop was held in April, 1992 on CMM v1.0, which was attended by about 200 software professionals. This version of the CMM, Version 1.1, is the result of the feedback from that workshop and ongoing feedback from the software community.

The CMM is the foundation for systematically building a set of tools, including a maturity questionnaire, which are useful in software process improvement. The essential point to remember is that the model, not a questionnaire, is the basis for improving the software process. This paper is intended to introduce the reader to CMM v1.1.

What is the Purpose of This Paper?

This paper provides a technical overview of the Capability Maturity Model for Software and reflects Version 1.1. Specifically, this paper describes the process maturity framework of five maturity levels, the structural components that comprise the CMM, how the CMM is used in practice, and future directions of the CMM. This paper serves as one of the best sources for understanding the CMM, and it should clear up some of the misconceptions associated with software process maturity as advocated by the SEI.

The SEI has worked with industry and government to refine and expand the model, and software organizations are encouraged to focus on the CMM rather than on the maturity questionnaire. The SEI has developed, and is developing, a suite of process products to encourage this focus. This paper [Paulk93a], in combination with the "Key Practices of the Capability Maturity Model, Version 1.1" [Paulk93b], comprises CMM v1.1. The "Key Practices of the Capability Maturity Model, Version 1.1" describes the key practices for each level of the CMM. This paper describes the principles underlying software process maturity and is intended to help software organizations use CMM v1.1 as a guide to improve the maturity of their software processes.

Who Should Read This Paper?

This paper presents an introduction to the CMM and its associated products. Therefore, anyone who is interested in learning about the CMM should read this paper. However, this paper assumes that the reader has some knowledge of, and experience in, developing and/or maintaining software, as well as an understanding of the problems that the software community faces today.

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,
- 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 know the risks of having a particular software organization perform the work of a contract,
- by the SEI as the basis for developing questions for the maturity questionnaire, and
- by instructors preparing teams to perform software process assessments or software capability evaluations.

How is This Paper Organized?

This paper has five chapters:

Chapter 1	Defines the concepts necessary to understand the CMM and the motivation and purpose behind it.

Chapter 2	Describes the five levels of the CMM and the principles that underlie them.
Chapter 3	Describes how the CMM is structured into key process areas, organized by common features, and described in terms of key practices.
Chapter 4	Provides a high-level overview of how the CMM provides guidance for software process assessments, software capability evaluations, and process improvement programs.
Chapter 5	Concludes by providing a description of future directions for the CMM and its related products.

What Are the Other CMM Products?

Although this paper can be read in isolation, it is designed to be the launching point for other products. This paper and the associated products help the reader understand and use the CMM. All of the CMM-based products have been, or will be, systematically derived from the model. At the time of this writing, most of these products are not available in their final form, although preliminary versions are in various stages of pilot testing and release.

The CMM-based set of products includes several diagnostic tools, which are used by software process assessment[1] and software capability evaluation[2] teams to identify strengths, weaknesses, and risks of an organization's software process. Probably the best known of these is the maturity questionnaire. The software process assessment and software capability evaluation methods and training also rely on the CMM.

The users of these products form a community dedicated to improving the maturity of their software process. The SEI will continue to work with the software community to enhance the model and its associated products.

How Do You Receive More Information?

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

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

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

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

To the Reader

SEI technical reports, such as this paper and the "Key Practices of the Capability Maturity Model, Version 1.1," are directly available from the Defense Technical Information Center (DTIC), the National Technical Information Service (NTIS), and Research Access Inc. (RAI). These documents can be obtained by contacting:

RAI: Research Access Inc.
3400 Forbes Avenue
Suite 302
Pittsburgh, PA 15213
Telephone: (800) 685-6510
FAX: (412) 682-6530

NTIS: National Technical Information Service
U.S. Department of Commerce
Springfield, VA 22161-2103
Telephone: (703) 487-4600

DTIC: Defense Technical Information Center
ATTN: FDRA Cameron Station
Alexandria, VA 22304-6145
Telephone: (703) 274-7633

SEI technical reports are also available via Internet. To use anonymous ftp from a Unix system on Internet:

```
ftp ftp.sei.cmu.edu[3]
login: anonymous
password: <your user id or any string>
cd pub/cmm
get READ.ME
get <files>
quit
```

The file READ.ME contains information on what files are available. Other SEI publications are available in a similar manner.

[3] The SEI ftp machine address is 128.237.2.179.

1 The Process Maturity Framework

After two decades of unfulfilled promises about productivity and quality gains from applying new software methodologies and technologies, industry and government organizations are realizing that their fundamental problem is the inability to manage the software process [DoD87]. The benefits of better methods and tools cannot be realized in the maelstrom of an undisciplined, chaotic project. In many organizations, projects are often excessively late and double the planned budget [Siegel90]. In such instances, the organization frequently is not providing the infrastructure and support necessary to help projects avoid these problems.

Even in undisciplined organizations, however, some individual software projects produce excellent results. When such projects succeed, it is generally through the heroic efforts of a dedicated team, rather than through repeating the proven methods of an organization with a mature software process. In the absence of an organization-wide software process, repeating results depends entirely on having the same individuals available for the next project. Success that rests solely on the availability of specific individuals provides no basis for long-term productivity and quality improvement throughout an organization. Continuous improvement can occur only through focused and sustained effort towards building a process infrastructure of effective software engineering and management practices.

1.1 Immature Versus Mature Software Organizations

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

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

In an immature organization, there is no objective basis for judging product quality or for solving product or process problems. Therefore, product quality is difficult to predict. Activities intended to enhance quality such as reviews and testing are often curtailed or eliminated when projects fall behind schedule.

On the other hand, a mature software organization possesses an organization-wide ability for managing software development and maintenance processes. The software process is accurately communicated to both existing staff and new employees, and work activities are carried out according to the planned process. The processes mandated are fit for use [Humphrey91b] and consistent with the way the work actually gets done. These defined processes are updated when necessary, and improvements are developed through controlled pilot-tests and/or cost benefit analyses. Roles and responsibilities within the defined process are clear throughout the project and across the organization.

In a mature organization, managers monitor the quality of the software products and customer satisfaction. There is an objective, quantitative basis for judging product quality and analyzing problems with the product and process. Schedules and budgets are based on historical performance and are realistic; the expected results for cost, schedule, functionality, and quality of the product are usually achieved. In general, a disciplined process is consistently followed because all of the participants understand the value of doing so, and the necessary infrastructure exists to support the process.

Capitalizing on these observations about immature and mature software organizations requires construction of a software process maturity

framework. This framework describes an evolutionary path from ad hoc, chaotic processes to mature, disciplined software processes. Without this framework, improvement programs may prove ineffective because the necessary foundation for supporting successive improvements has not been established. The software process maturity framework emerges from integrating the concepts of software process, software process capability, software process performance, and software process maturity, all of which are defined in succeeding paragraphs.

1.2 Fundamental Concepts Underlying Process Maturity

According to Webster's dictionary, a *process* is "a system of operations in producing something ... a series of actions, changes, or functions that achieve an end or result." The IEEE defines a process as "a sequence of steps performed for a given purpose" [IEEE-STD-610]. A *software process* can be defined as a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, and user manuals). As an organization matures, the software process becomes better defined and more consistently implemented throughout the organization.

Software process capability describes the range of expected results that can be achieved by following a software process. The software process capability of an organization provides one means of predicting the most likely outcomes to be expected from the next software project the organization undertakes.

Software process performance represents the actual results achieved by following a software process. Thus, software process performance focuses on the results achieved, while software process capability focuses on results expected. Based on the attributes of a specific project and the context within which it is conducted, the actual performance of the project may not reflect the full process capability of the organization; i.e., the capability of the

project is constrained by its environment. For instance, radical changes in the application or technology undertaken may place a project' s staff on a learning curve that causes their project's capability, and performance, to fall short of the organization's full process capability.

Software process maturity is the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective. Maturity implies a potential for growth in capability and indicates both the richness of an organization's software process and the consistency with which it is applied in projects throughout the organization. The software process is well-understood throughout a mature organization, usually through documentation and training, and the process is continually being monitored and improved by its users. The capability of a mature software process is known. Software process maturity implies that the productivity and quality resulting from an organization's software process can be improved over time through consistent gains in the discipline achieved by using its software process.

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

1.3 Overview of the Capability Maturity Model

Although software engineers and managers often know their problems in great detail, they may disagree on which improvements are most important. Without an organized strategy for improvement, it is difficult to achieve consensus between management and the professional staff on what improvement activities to undertake first. To achieve lasting results from process improvement efforts, it is necessary to design an evolutionary path

that increases an organization's software process maturity in stages. The software process maturity framework [Humphrey 87a] orders these stages so that improvements at each stage provide the foundation on which to build improvements undertaken at the next stage. Thus, an improvement strategy drawn from a software process maturity framework provides a roadmap for continuous process improvement. It guides advancement and identifies deficiencies in the organization; it is not intended to provide a quick fix for projects in trouble.

The Capability Maturity Model for Software provides software organizations with guidance on how to gain control of their processes for developing and maintaining software and how to evolve toward a culture of software engineering and management excellence. The CMM was designed to guide software organizations in selecting process improvement strategies by determining current process maturity and identifying the few issues most critical to software quality and process improvement. By focusing on a limited set of activities and working aggressively to achieve them, an organization can steadily improve its organization-wide software process to enable continuous and lasting gains in software process capability.

The staged structure of the CMM is based on principles of product quality that have existed for the last sixty years. In the 1930s, Walter Shewart, promulgated the principles of statistical quality control. His principles were further developed and successfully demonstrated in the work of W. Edwards Deming [Deming86] and Joseph Juran [Juran88, Juran89]. These principles have been adapted by the SEI into a maturity framework that establishes a project management and engineering foundation for quantitative control of the software process, which is the basis for continuous process improvement.

The maturity framework into which these quality principles have been adapted was first inspired by Philip Crosby of in his book *Quality is Free* [Crosby79]. Crosby's quality management maturity grid describes five evolutionary stages in adopting quality practices. This maturity framework was adapted to the software process by Ron Radice and his colleagues,

working under the direction of Watts Humphrey at IBM [Radice85]. Humphrey brought this maturity framework to the Software Engineering Institute in 1986, added the concept of maturity levels, and developed the foundation for its current use throughout the software industry.

Early versions of Humphrey's maturity framework are described in SEI technical reports [Humphrey87a, Humphrey87b], papers [Humphrey88], and in his book, *Managing the Software Process* [Humphrey89]. A preliminary maturity questionnaire [Humphrey87b] was released in 1987 as a tool to provide organizations with a way to characterize the maturity of their software processes. Two methods, software process assessment and software capability evaluation, were developed to appraise software process maturity in 1987. Since 1990, the SEI, with the help of many people from government and industry, has further expanded and refined the model based on several years of experience in its application to software process improvement.

2 The Five Levels of Software Process Maturity

Continuous process improvement is based on many small, evolutionary steps rather than revolutionary innovations [Imai86]. The CMM provides a framework for organizing these evolutionary steps into five maturity levels that lay successive foundations for continuous process improvement. These five maturity levels define an ordinal scale for measuring the maturity of an organization's software process and for evaluating its software process capability. The levels also help an organization prioritize its improvement efforts.

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

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

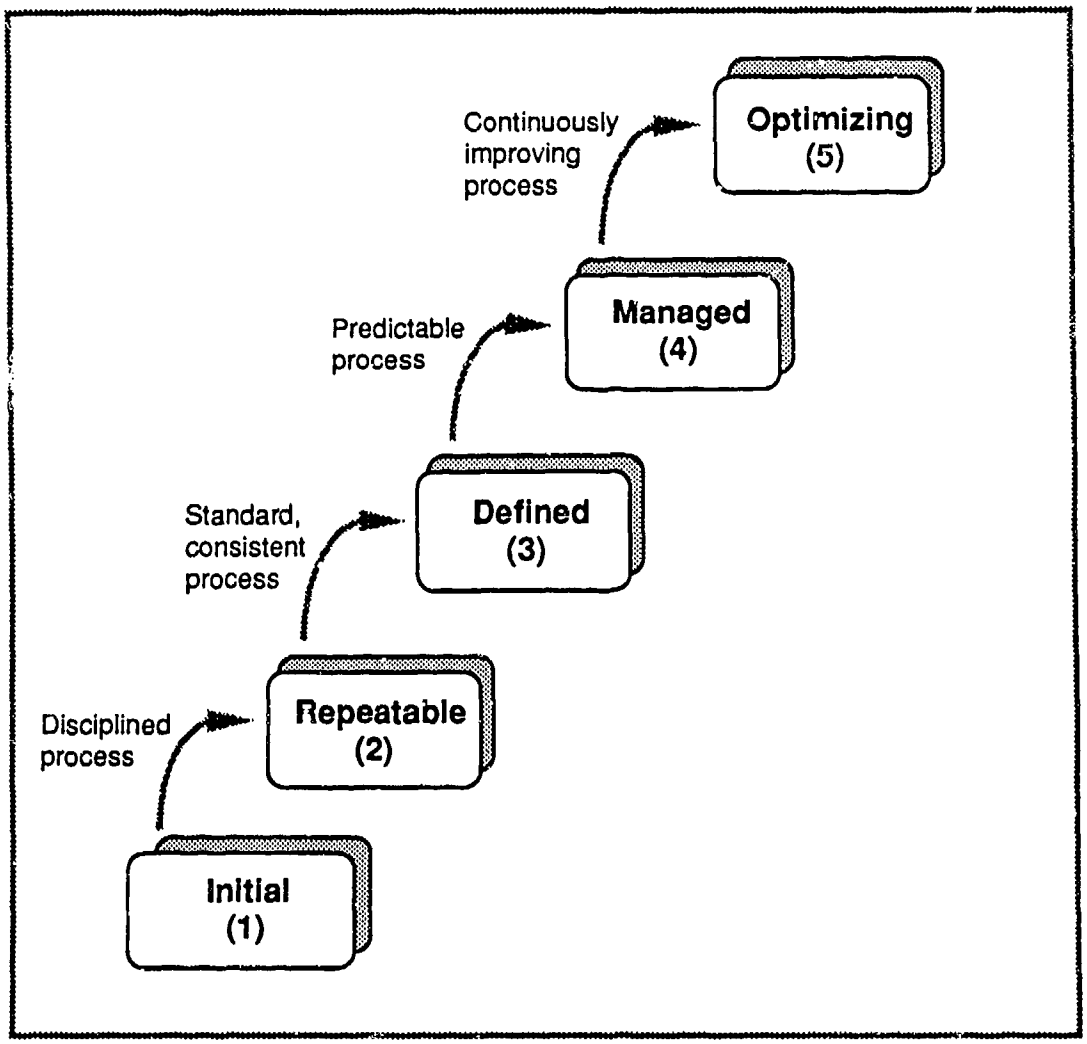

Figure 2.1 The Five Levels of Software Process Maturity

The following characterizations of the five maturity levels highlight the primary process changes made at each level:

1) Initial — The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

2) *Repeatable* — Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

3) *Defined* — The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

4) *Managed* — Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

5) *Optimizing* — Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

2.1 Behavioral Characterization of the Maturity Levels

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

2.1.1 Level 1 - The Initial Level

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

2.1.2 Level 2 - The Repeatable Level

At the Repeatable Level, policies for managing a software project and procedures to implement those policies are established. Planning and managing new projects is based on experience with similar projects. 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.

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.

2.1.3 Level 3 - The Defined Level

At the Defined Level, the standard process for developing and maintaining software across the organization is documented, including both software engineering and management processes, and these processes are integrated into a coherent whole. This standard process is referred to throughout the CMM as the organization's standard software process. Processes established at Level 3 are used (and changed, as appropriate) to help the software managers and technical staff perform more effectively. The organization exploits effective software engineering practices when standardizing its software processes. There is a group that is responsible for the

organization's software process activities, e.g., a software engineering process group, or SEPG [Fowler90]. An organization-wide training program is implemented to ensure that the staff and managers have the knowledge and skills required to fulfill their assigned roles.

Projects tailor the organization's standard software process to develop their own defined software process, which accounts for the unique characteristics of the project. This tailored process is referred to in the CMM as the project's defined software process. A defined software process contains a coherent, integrated set of well-defined software engineering and management processes. A well-defined process can be characterized as including readiness criteria, inputs, standards and procedures for performing the work, verification mechanisms (such as peer reviews), outputs, and completion criteria. Because the software process is well defined, management has good insight into technical progress on all projects.

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

2.1.4 Level 4 - The Managed Level

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

These measurements establish the quantitative foundation for evaluating the projects' software processes and products.

Projects achieve control over their products and processes by narrowing the variation in their process performance to fall within acceptable quantitative boundaries. Meaningful variations in process performance can be distinguished from random variation (noise), particularly within established product lines. The risks involved in moving up the learning curve of a new application domain are known and carefully managed.

The software process capability of Level 4 organizations can be summarized as 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.

2.1.5 Level 5 - The Optimizing Level

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

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

The software process capability of Level 5 organizations can be characterized as continuously improving because Level 5 organizations are continuously striving to improve the range of their process capability, thereby improving the process performance of their projects. Improvement occurs both by incremental advancements in the existing process and by innovations using new technologies and methods.

2.2 Understanding the Maturity Levels

The CMM is a descriptive model in the sense that it describes essential (or key) attributes that would be expected to characterize an organization at a particular maturity level. It is a normative model in the sense that the detailed practices characterize the normal types of behavior that would be expected in an organization doing large-scale projects in a government contracting context. The intent is that the CMM is at a sufficient level of abstraction that it does not unduly constrain how the software process is implemented by an organization; it simply describes what the essential attributes of a software process would normally be expected to be.

In any context in which the CMM is applied, a reasonable interpretation of the practices should be used. The CMM must be appropriately interpreted, using informed professional judgment, when the business environment of the organization differs significantly from that of a large contracting organization.

The CMM is not prescriptive; it does not tell an organization how to improve. The CMM describes an organization at each maturity level without prescribing the specific means for getting there. It can take several years to move from Level 1 to Level 2, and moving between the other levels will usually take on the order of two years.

Software process improvement occurs within the context of the organization's strategic plans and business objectives, its organizational structure, the technologies in use, its social culture, and its management system. The CMM focuses on the process aspects of a Total Quality Management effort; successful process improvement implies that aspects outside the scope of software process are also addressed (e.g., the people issues involved with changing the organizational culture that enable the process improvements to be implemented and institutionalized).

2.2.1 Understanding the Initial Level

Although Level 1 organizations are frequently characterized as having ad hoc, even chaotic, processes, they frequently develop products that work, even though they may be over the budget and schedule. Success in Level 1 organizations depends on the competence and heroics of the people in the organization. Selecting, hiring, developing, and/or retaining competent people are significant issues for organizations at all levels of maturity, but they are largely outside the scope of the CMM.

2.2.2 Understanding the Repeatable and Defined Levels

As projects grow in size and complexity, attention shifts from technical issues to organizational and managerial issues – the focus of process maturity [Siegel90, DoD87, GAO-92-48]. Process enables people to work more effectively by incorporating the lessons learned by the best staff into documented processes, building the skills needed to perform those processes effectively (usually via training), and continually improving by learning from the people performing the job.

To achieve Level 2, management must focus on its own processes to achieve a disciplined software process. Level 2 provides the foundation for Level 3 because the focus is on management acting to improve its processes

before tackling technical and organizational issues at Level 3. Management establishes a leadership position in achieving Level 2 by documenting and following project management processes.

Processes may differ between projects in a Level 2 organization; the organizational requirement for achieving Level 2 is that there are policies that guide the projects in establishing the appropriate management processes. Documented procedures provide the foundation for consistent processes that can be institutionalized across the organization, with the aid of training and software quality assurance.

Level 3 builds on this project management foundation by defining, integrating, and documenting the entire software process. Integration in this case means that the outputs of one task flow smoothly into the inputs of the next task. When there are mismatches between tasks, they are identified and addressed in the planning stages of the software process, rather than when they are encountered while enacting the process. One of the challenges of Level 3 is to build processes that empower the individuals doing the work without being overly rigid [Humphrey 91b].

2.2.3 Understanding the Managed and Optimizing Levels

Maturity Levels 4 and 5 are relatively unknown territory for the software industry. There are only a few examples of Level 4 and 5 software projects and organizations [Humphrey91a, Kitson92]. There are too few to draw general conclusions about the characteristics of Level 4 and 5 organizations. The characteristics of these levels have been defined by analogy with other industries and the few examples in the software industry exhibiting this level of process capability.

Many characteristics of Levels 4 and 5 are based on the concepts of statistical process control as exemplified in Figure 2.2. The Juran Trilogy Diagram [Juran88] illustrates the primary objectives of process management.

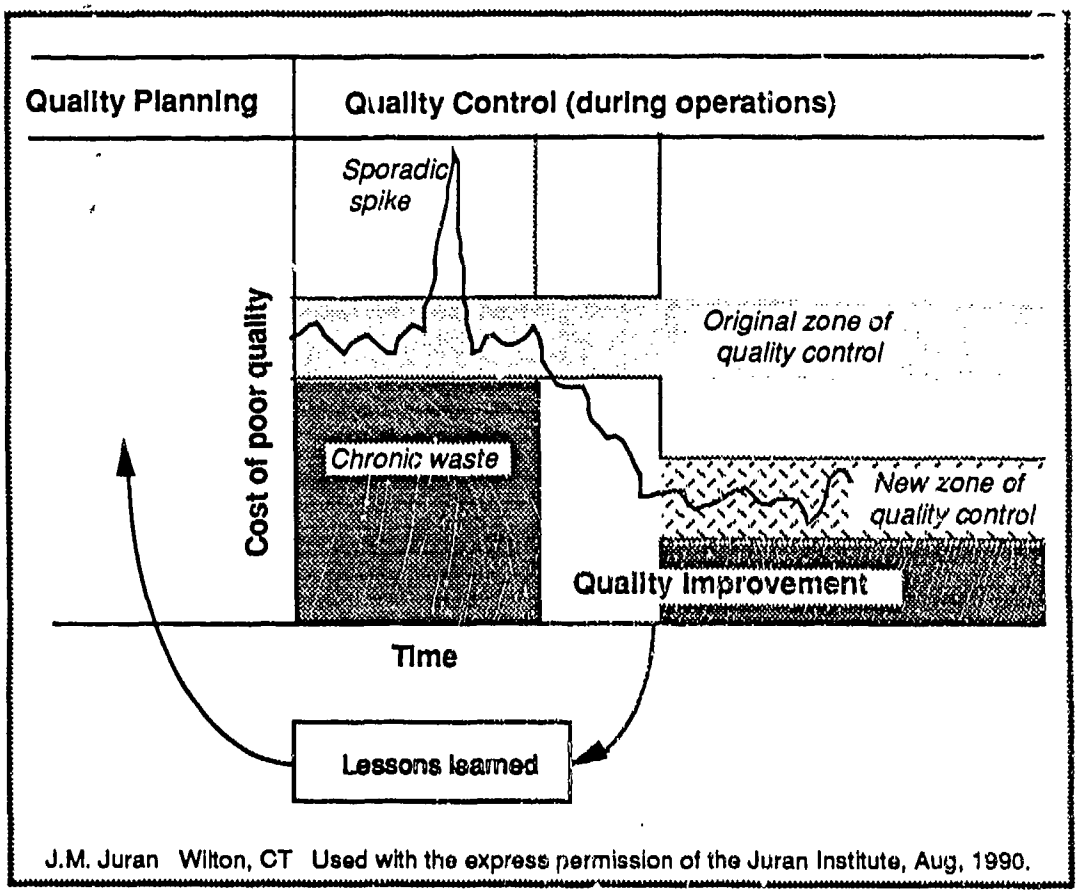

Figure 2.2 The Juran Trilogy Diagram: Quality Planning, Quality Control, and Quality Improvement

Juran breaks quality management into three basic managerial processes [Juran88]. The purpose of quality planning is to provide the operating forces, i.e., the software producers, with the means of producing products that can meet customer needs. The operating forces produce the product, but some rework must be done because of quality deficiencies. This waste is chronic because the process was planned that way; quality control is carried out to prevent things from getting worse. Sporadic spikes in the process, as shown in Figure 2.2, represent fire fighting activities. Chronic waste provides an opportunity for improvement; seizing that opportunity is referred to as quality improvement.

The first responsibility, and the focus of Level 4, is process control. The software process is managed so that it operates stably within a zone of quality control. There is inevitably some chronic waste, and there may be spikes in the measured results that need to be controlled, but the system is generally stable overall. This is where the concept of controlling special causes of variation comes into play. Because the process is both stable and measured, when some exceptional circumstance occurs, the "special cause" of the variation can be identified and addressed.

The second responsibility, and the focus of Level 5, is continuous process improvement. The software process is changed to improve quality, and the zone of quality control moves. A new baseline for performance is established that reduces chronic waste. The lessons learned in improving such a process are applied in planning future processes. This is where the concept of addressing common causes of variation comes to the fore. There is chronic waste, in the form of rework, in any system simply due to random variation. Waste is unacceptable; organized efforts to remove waste result in changing the system, i.e., improving the process by changing "common causes" of inefficiency to prevent the waste from occurring.

It is anticipated that organizations reaching the highest maturity levels of the CMM would have a process that is capable of producing extremely reliable software within predictable cost and schedule limits. As understanding of the higher maturity levels grows, the existing key process areas will be refined, and others may be added to the model. The CMM is derived from ideas about process that were inspired in manufacturing, but software processes are not dominated by replication issues like a manufacturing process is. The software process is dominated by design issues and is a knowledge-intensive activity [Curtis88].

2.3 Visibility Into the Software Process

Software engineers have detailed insight into the state of a project because they have first-hand information on project status and performance. However, on large projects their insight usually is drawn only from their personal experience in their area of responsibility. Those outside the project without first-hand exposure, such as senior managers, lack visibility into the project's processes and rely on periodic reviews for the information they require to monitor progress. Figure 2.3 illustrates the level of visibility into project status and performance afforded to management at each level of process maturity. Each succeeding maturity level incrementally provides better visibility into the software process.

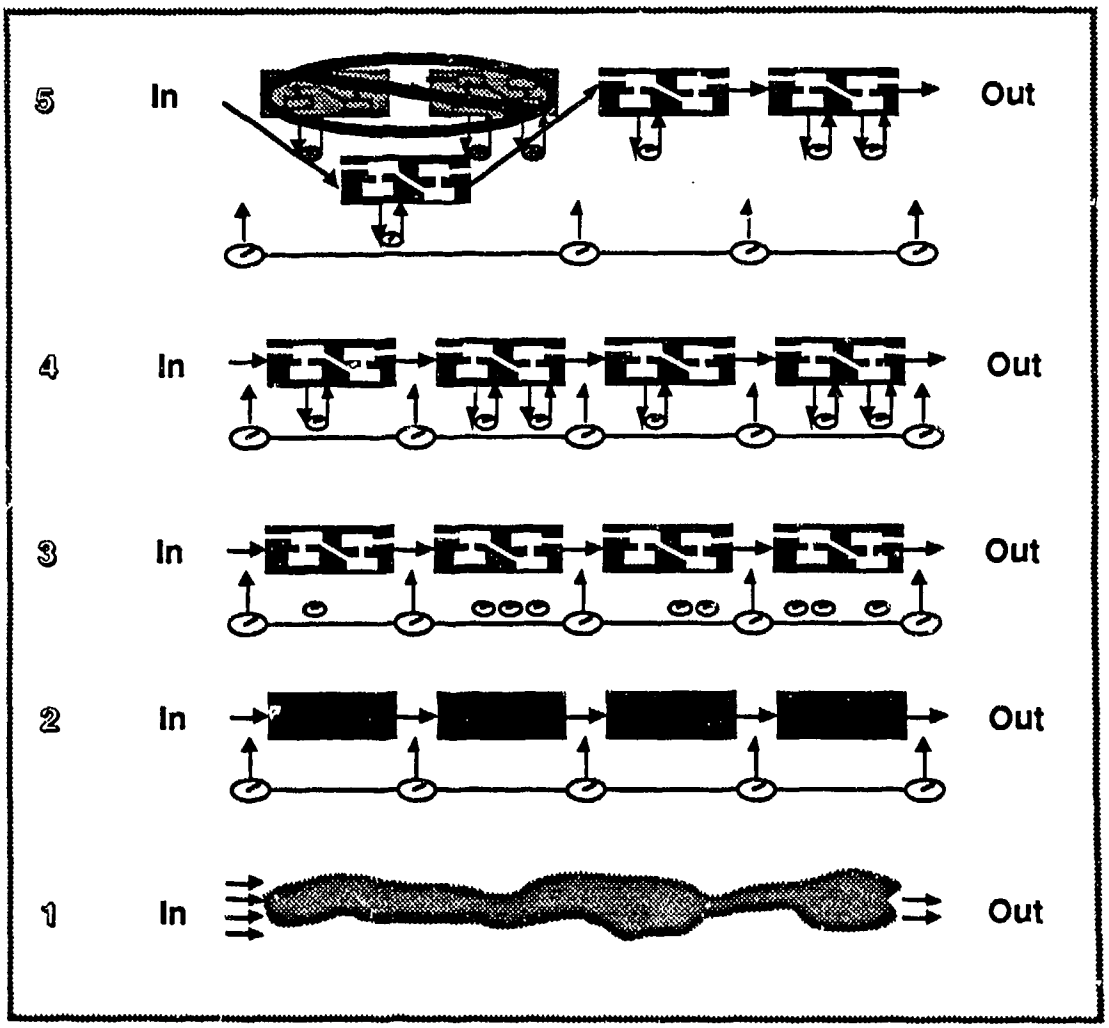

Figure 2.3 A Management View of Visibility Into the Software Process at Each Maturity Level

At Level 1, the software process is an amorphous entity – a black box – and visibility into the project's processes is limited. Since the staging of activities is poorly defined, managers have an extremely difficult time establishing the status of the project's progress and activities.[4]

[4] This leads to the Ninety-Ninety Rule: 90% of the project is complete 90% of the time.

Requirements flow into the software process in an uncontrolled manner, and a product results. Software development is frequently viewed as black magic, especially by managers who are unfamiliar with software.

At Level 2, the customer requirements and work products are controlled, and basic project management practices have been established. These management controls allow visibility into the project on defined occasions. The process of building the software can be viewed as a succession of black boxes that allows management visibility at transition points as activity flows between boxes (project milestones). Even though management may not know the details of what is happening in the box, the products of the process and checkpoints for confirming that the process is working are identified and known. Management reacts to problems as they occur.

At Level 3, the internal structure of the boxes, i.e., the tasks in the project's defined software process, is visible. The internal structure represents the way the organization's standard software process has been applied to specific projects. Both managers and engineers understand their roles and responsibilities within the process and how their activities interact at the appropriate level of detail. Management proactively prepares for risks that may arise. Individuals external to the project can obtain accurate and rapid status updates because defined processes afford great visibility into project activities.

At Level 4, the defined software processes are instrumented and controlled quantitatively. Managers are able to measure progress and problems. They have an objective, quantitative basis for making decisions. Their ability to predict outcomes grows steadily more precise as the variability in the process grows smaller.

At Level 5, new and improved ways of building the software are continually tried, in a controlled manner, to improve productivity and quality. Disciplined change is a way of life as inefficient or defect-prone activities are identified and replaced or revised. Insight extends beyond existing processes

and into the effects of potential changes to processes. Managers are able to estimate and then track quantitatively the impact and effectiveness of change.

2.4 Process Capability and the Prediction of Performance

The maturity of an organization's software process helps to predict a project's ability to meet its goals. Projects in Level 1 organizations experience wide variations in achieving cost, schedule, functionality, and quality targets. As illustrated in Figure 2.4, three improvements in meeting targeted goals are observed as the organization's software process matures.

First, as maturity increases, the difference between targeted results and actual results decreases across projects. For instance, if ten projects of the same size were targeted to be delivered on May 1, then the average date of their delivery would move closer to May 1 as the organization matures. Level 1 organizations often miss their originally scheduled delivery dates by a wide margin, whereas Level 5 organizations should be able to meet targeted dates with considerable accuracy. This is because Level 5 organizations use a carefully constructed software process operating within known parameters, and the selection of the target date is based on the extensive data they possess about their process and on their performance in applying it. (This is illustrated in Figure 2.4 by how much of the area under the curve lies to the right of the target line.)

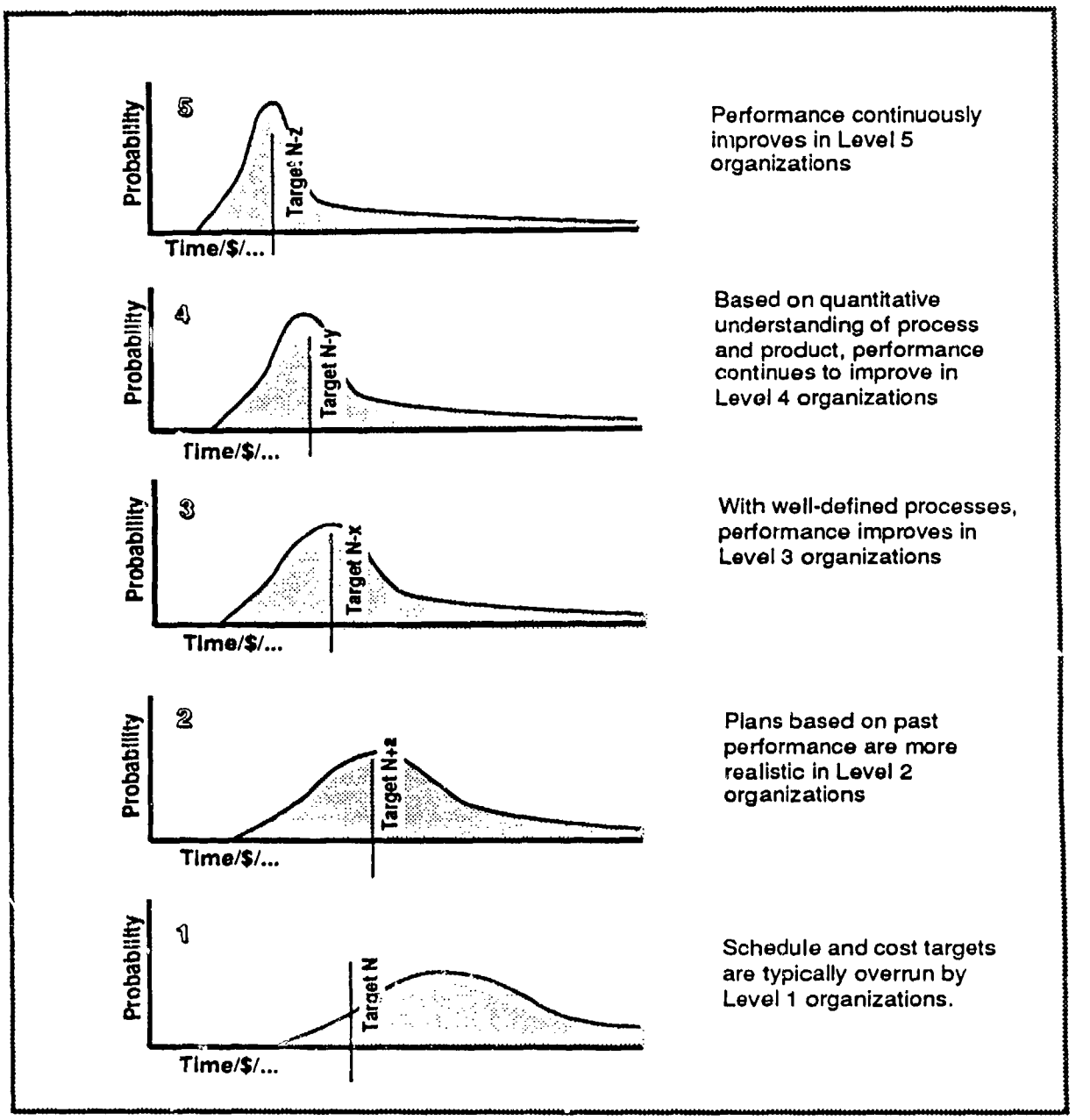

Figure 2.4 Process Capability as Indicated by Maturity Level

Second, as maturity increases, the variability of actual results around targeted results decreases. For instance, in Level 1 organizations delivery dates for projects of similar size are unpredictable and vary widely. Similar projects in a Level 5 organization, however, will be delivered within a much smaller range. This narrowed variation occurs at the highest maturity levels because virtually all projects are performing within controlled parameters approaching the organization's process capability for cost, schedule, functionality, and quality. (This is illustrated in Figure 2.4 by how much of the area under the curve is concentrated near the target line.)

Third, targeted results improve as the maturity of the organization increases. That is, as a software organization matures, costs decrease, development time becomes shorter, and productivity and quality increase. In a Level 1 organization, development time can be quite long because of the amount of rework that must be performed to correct mistakes. In contrast, Level 5 organizations use continuous process improvement and defect prevention techniques to increase process efficiency and eliminate costly rework, allowing development time to be shortened. (This is illustrated in Figure 2.4 by the horizontal displacement of the target line from the origin.)

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

2.5 Skipping Maturity Levels

The maturity levels in the CMM describe the characteristics of an organization at a maturity level. Each level builds a foundation for succeeding levels to leverage for implementing processes effectively and efficiently. Organizations can, however, profitably use processes described at a higher maturity level than they are. Engineering processes, such as requirements analysis, design, code, and test, are not discussed in the CMM until Level 3, yet even Level 1 organizations must perform these activities. A Level 1 or Level 2 organization may be able to perform peer reviews (Level 3), do Pareto analysis (Level 4), or pilot new technologies (Level 5) profitably. When prescribing what steps an organization should take to move from Level 1 to Level 2, frequently one of the recommendations is to establish a software engineering process group, which is an attribute of Level 3 organizations. Although measurement is the focus of Level 4, it is also an integral part of the lower maturity levels.

These processes cannot reach their full potential, however, until the proper foundation is laid. Peer reviews cannot be fully effective, for example, unless they are consistently implemented, even when fires threaten the project. The maturity levels describe the problems that predominate at a level. The dominant problems of a Level 1 organization are managerial; other problems tend to be masked by the difficulties in planning and managing software projects.

Skipping levels is counterproductive because each level forms a necessary foundation from which to achieve the next level. The CMM identifies the levels through which an organization must evolve to establish a culture of software engineering excellence. Processes without the proper foundation fail at the very point they are needed most – under stress – and they provide no basis for future improvement.

A Level 1 organization that is trying to implement a defined process (Level 3) before it has established a repeatable process (Level 2) is usually unsuccessful because project managers are overwhelmed by schedule and cost pressures. This is the fundamental reason for focusing on management processes before engineering processes. It may seem easier to define and implement an engineering process than a management process (especially in the eyes of technical people), but without management discipline, the engineering process is sacrificed to schedule and cost pressures [Humphrey88].

An organization that is trying to implement a managed process (Level 4) without the foundation of a defined process is usually unsuccessful because there is no common basis for interpreting measurements without defined processes. While data can be collected for individual projects, few of the measurements have significant meaning across projects, and they do not significantly increase organizational understanding of the software process. It is difficult to identify meaningful measurements in the absence of defined processes because of the variation in the processes being measured.

An organization that is trying to implement an optimizing process (Level 5) without the foundation of a managed process (Level 4) is likely to fail because of a lack of understanding of the impact of process changes. Without controlling the process within statistically narrow boundaries (small variations in process measures), there is too much noise in the data to determine objectively whether a specific process improvement has an effect. Decisions can degenerate into religious wars because little quantitative foundation exists for making rational, informed decisions.

The process improvement effort should focus on the needs of the organization in the context of its business environment. The ability to implement processes from higher maturity levels does not imply that maturity levels can be skipped.

3 Operational Definition of the Capability Maturity Model

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

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

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

3.1 Internal Structure of the Maturity Levels

Each maturity level has been decomposed into constituent parts. With the exception of Level 1, the decomposition of each maturity level ranges from abstract summaries of each level down to their operational definition in the key practices, as shown in Figure 3.1. Each maturity level is composed of several key process areas. Each key process area is organized into five

sections called common features. The common features specify the key practices that, when collectively addressed, accomplish the goals of the key process area.

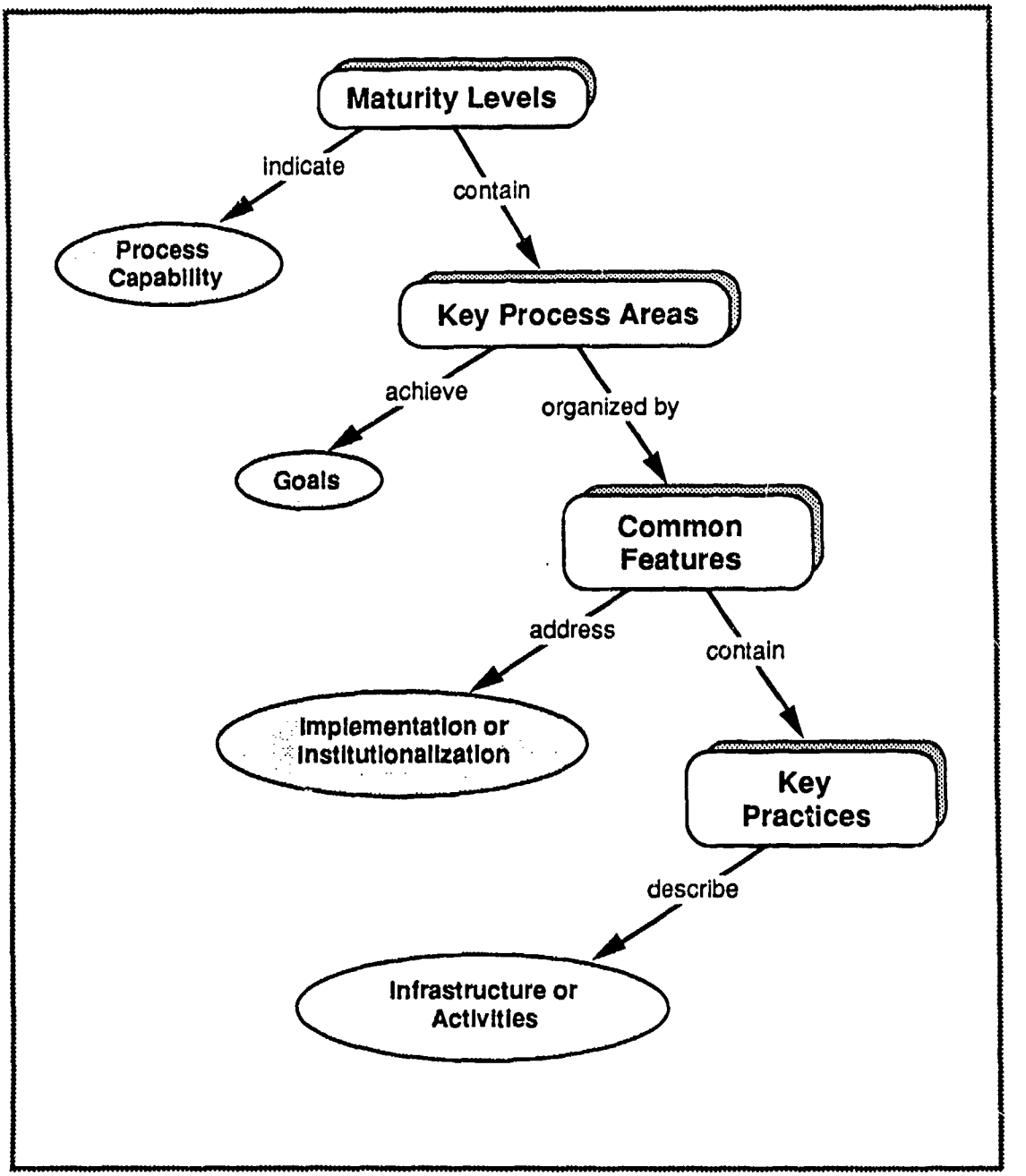

Figure 3.1 The CMM Structure

3.2 Maturity Levels

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

3.3 Key Process Areas

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

Each *key process area* identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing process capability. The key process areas have been defined to reside at a single maturity level as shown in Figure 3.2. The path to achieving the goals of a key process area may differ across projects based on differences in application domains or environments. Nevertheless, all the goals of a key process area must be achieved for the organization to satisfy that key process area. When the goals of a key process area are accomplished on a continuing basis across projects, the organization can be said to have institutionalized the process capability characterized by the key process area.

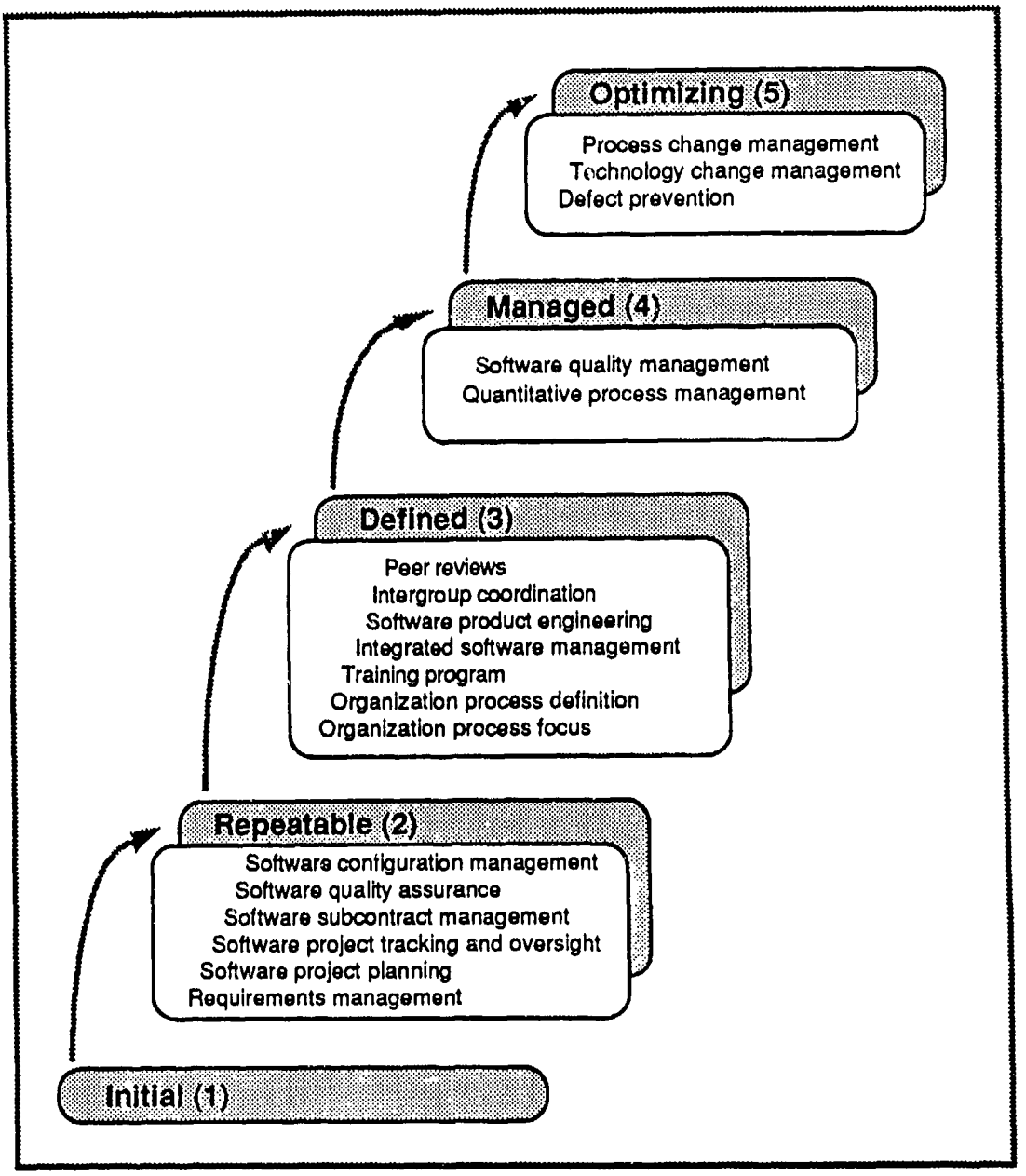

Figure 3.2 The Key Process Areas by Maturity Level

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

Although other issues affect process performance, the key process areas were identified because of their effectiveness in improving an organization's software process capability. They may be considered the requirements for achieving a maturity level. Figure 3.2 displays the key process areas for each maturity level. To achieve a maturity level, the key process areas for that level must be satisfied. To satisfy a key process area, each of the goals for the key process area must be satisfied. 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.[5]

The specific practices to be executed in each key process area will evolve as the organization achieves higher levels of process maturity. For instance, many of the project estimating capabilities described in the Software Project Planning key process area at Level 2 must evolve to handle the additional project data available at Levels 3, 4, and 5. Integrated Software Management at Level 3 is the evolution of Software Project Planning and Software Project Tracking and Oversight at Level 2 as the project is managed using a defined software process.

The key process areas of the CMM represent one way of describing how organizations mature. These key process areas were defined based on many years of experience in software engineering and management and over five years of experience with software process assessments and software capability evaluations.

[5] For a listing of the goals for each key process area, refer to Appendix A.

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.

- 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.

- 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

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:

- 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.

3.4 Common Features

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

Commitment to Perform

Commitment to Perform describes the actions the organization must take to ensure that the process is established and will endure. Commitment to Perform typically involves establishing organizational policies and senior management sponsorship.

Ability to Perform

Ability to Perform describes the preconditions that must exist in the project or organization to implement the software process competently. Ability to Perform typically involves resources, organizational structures, and training.

Activities Performed

Activities Performed describes the roles and procedures necessary to implement a key process area. Activities Performed typically involve establishing plans and procedures, performing the work, tracking it, and taking corrective actions as necessary.

Measurement and Analysis

Measurement and Analysis describes the need to measure the process and analyze the measurements. Measurement and Analysis typically includes examples of the measurements that could be taken to determine the status and effectiveness of the Activities Performed.

Verifying Implementation

Verifying Implementation describes the steps to ensure that the activities are performed in compliance with the process that has been established. Verification typically encompasses reviews and audits by management and software quality assurance.

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

3.5 Key Practices

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

Each key practice consists of a single sentence, often followed by a more detailed description, which may include examples and elaboration. These key practices, also referred to as the top-level key practices, state the fundamental policies, procedures, and activities for the key process area. The components of the detailed description are frequently referred to as subpractices. Figure 3.3 provides an example of the structure underlying a key practice for the Software Project Planning key process area.

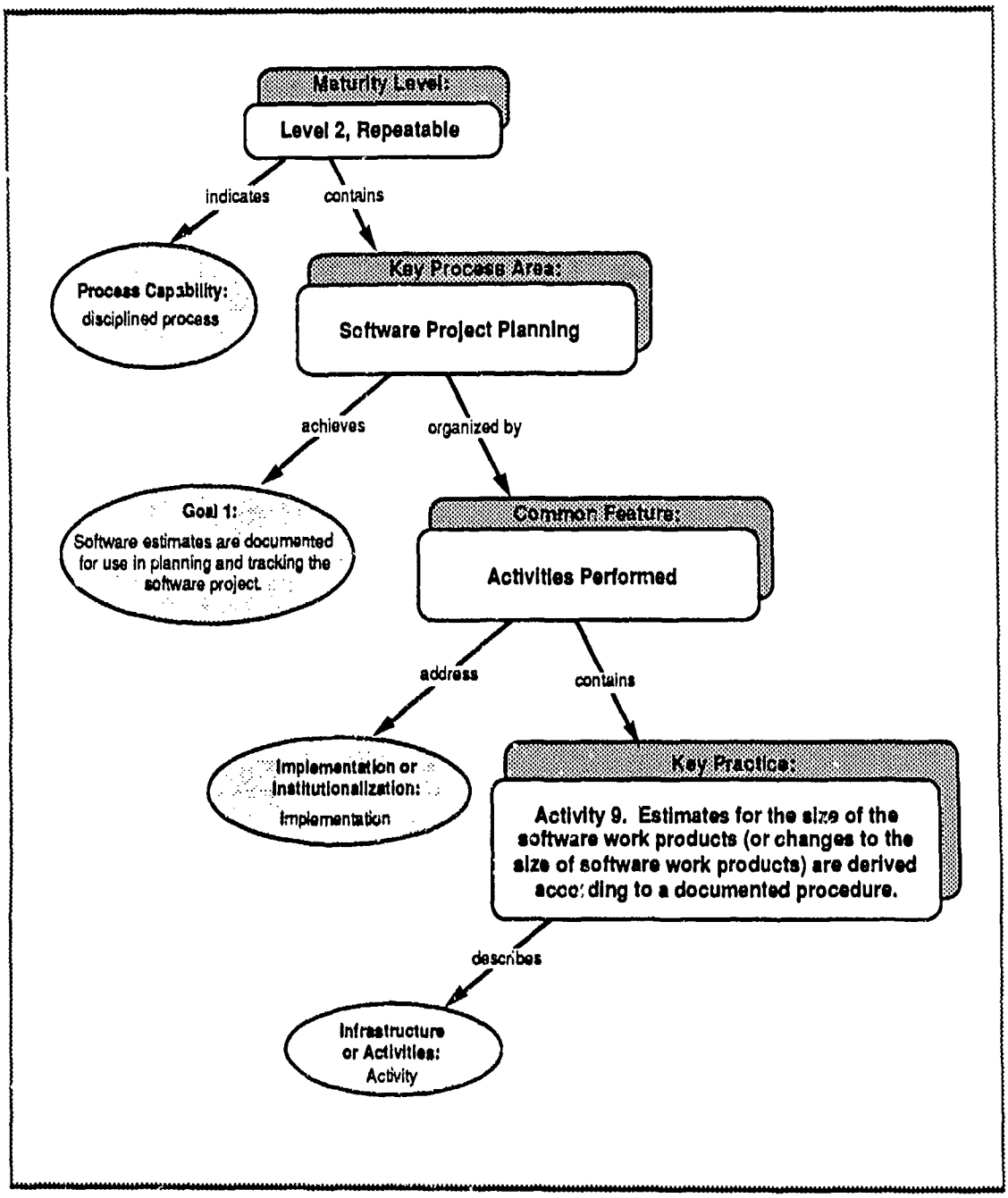

Figure 3.3 Building the CMM Structure: An Example of a Key Practice

As illustrated in Figure 3.3, to ensure consistent accomplishment of the goal of documenting plans for planning and tracking the project, the organization must establish a documented procedure for deriving estimates of software size. If these estimates are not developed from a documented procedure, they may vary wildly as differences in sizing assumptions are never surfaced. The detailed description of what would be expected in such a procedure includes using historical size data, documenting assumptions, and reviewing the estimates. These criteria guide the judgment of whether a reasonable size estimating procedure is followed.

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

4 Using the CMM

The CMM establishes a set of public in available criteria describing the characteristics of mature software organizations. These criteria can be used by organizations to improve their processes for developing and maintaining software, or by government or commercial organizations to evaluate the risks of contracting a software project to a particular company.

This chapter focuses on two SEI-developed methods for appraising the maturity of an organization's execution of the software process: software process assessment and software capability evaluation.

- *Software process assessments* are used to determine the state of an organization's current software process, to determine the high-priority software process-related issues facing an organization, and to obtain the organizational support for software process improvement.
- *Software capability evaluations* are used to identify contractors who are qualified to perform the software work or to monitor the state of the software process used on an existing software effort.

This overview is not sufficient by itself for readers to conduct either an assessment or evaluation. Anyone wishing to apply the CMM through these methods should request further information on assessment and evaluation training.

The CMM is a common foundation for both software process assessments and software capability evaluations. The purpose of the methods are quite different, however, and there are significant differences in the specific methods used. Both are based on the model and the products derived from it. The CMM, in conjunction with the CMM-based products, enables the methods to be applied reliably and consistently.

4.1 Software Process Assessment and Software Capability Evaluation Methods

Software process assessments focus on identifying improvement priorities within an organization's own software process. Assessment teams use the CMM to guide them in identifying and prioritizing findings. These findings, along with guidance provided by the key practices in the CMM, are used (by a software engineering process group, for example) to plan an improvement strategy for the organization.

Software capability evaluations are focused on identifying the risks associated with a particular project or contract for building high-quality software on schedule and within budget. During the acquisition process, software capability evaluations may be performed on bidders. The findings of the evaluation, as structured by the CMM, may be used to identify the risks in selecting a particular contractor. Evaluations may also be performed on existing contracts to monitor their process performance, with the intent of identifying potential improvements in the software process of the contractor.

The CMM establishes a common frame of reference for performing software process assessments and software capability evaluations. Although the two methods differ in purpose, the methods use the CMM as a foundation for appraising software process maturity. Figure 4.1 provides a summary description of the common steps in assessments and evaluations.

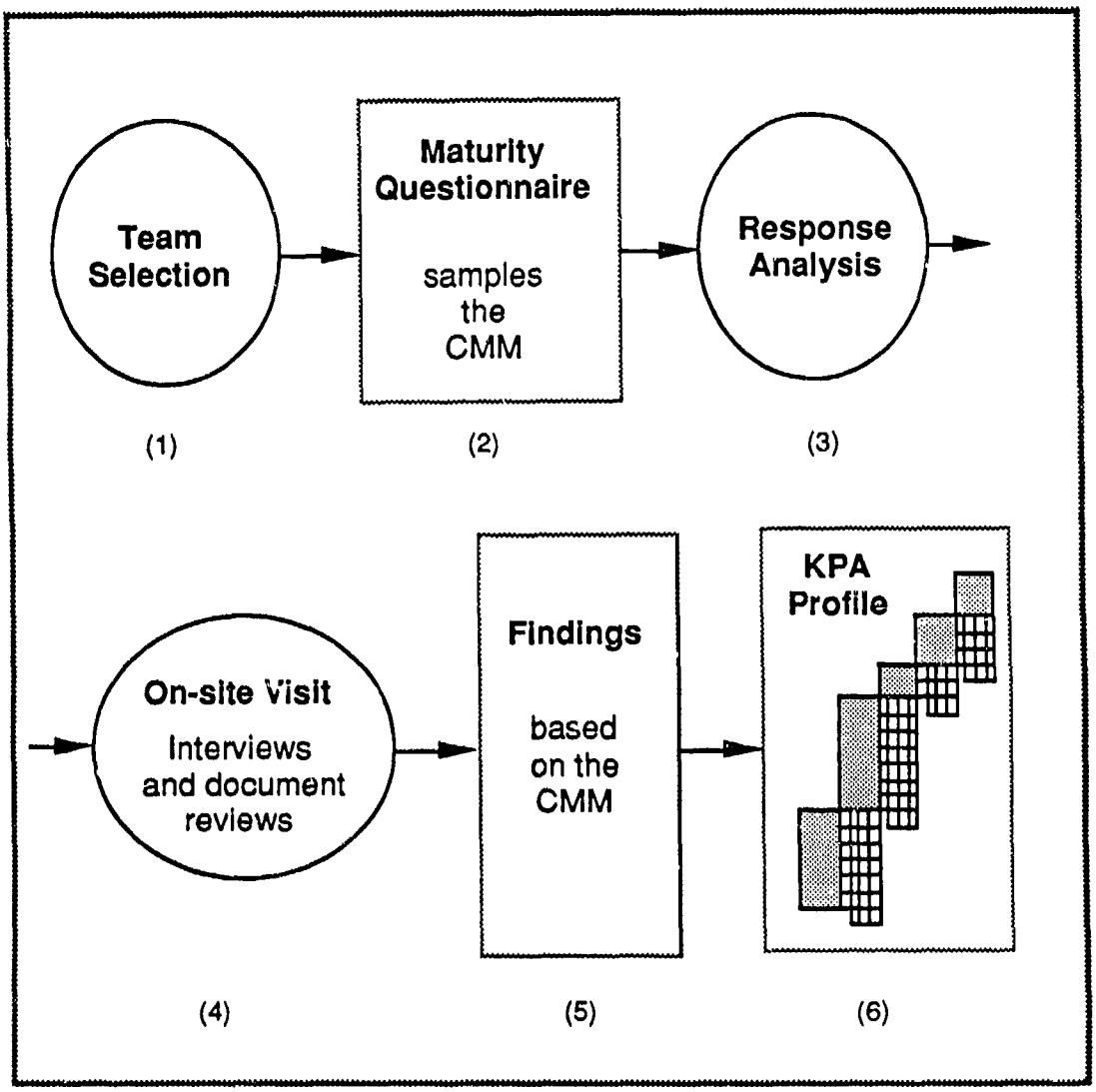

Figure 4.1 Common Steps in Software Process Assessments and Software Capability Evaluations

The first step in is to select a team. This team should be trained in the fundamental concepts of the CMM as well as the specifics of the assessment or evaluation method. The members of the team should be professionals knowledgeable in software engineering and management.

The second step is to have representatives from the site to be assessed or evaluated complete the maturity questionnaire and other diagnostic instruments. Once this activity is completed, the assessment or evaluation team performs a response analysis (step 3), which tallies the responses to the questions and identifies those areas where further exploration is warranted. The areas to be investigated correspond to the CMM key process areas.

The team is now ready to visit the site being assessed or evaluated (step 4). Beginning with the results of the response analysis, the team conducts interviews and reviews documentation to gain an understanding of the software process followed by the site. The key process areas and key practices in the CMM guide the team members in questioning, listening, reviewing, and synthesizing the information received from the interviews and documents. The team applies professional judgment in deciding whether the site's implementation of the key process areas satisfies the relevant key process area goals.[6] When there are clear differences between the key practices in the CMM and the site's practices, the team must document its rationale for judging that key process area.

At the end of the on-site period, the team produces a list of findings (step 5) that identifies the strengths and weaknesses of the organization's software process. In a software process assessment, the findings become the basis for recommendations for process improvement; in a software capability evaluation, the findings become part of the risk analysis performed by the acquisition agency.

Finally, the team prepares a key process area profile (step 6) that shows the areas where the organization has, and has not, satisfied the goals of the key process areas. A key process area can be satisfied and still have associated findings, provided the findings do not identify major problems that inhibit achieving any goals of the key process areas.

[6] These judgements may have to take place without complete information when company proprietary or security issues may be involved.

In summary, the software process assessment and software capability evaluation methods both:

- use the maturity questionnaire as a springboard for the on-site visit,
- use the CMM as a map that guides the on-site investigation,
- develop findings that identify software process strengths and weaknesses in terms of the key process areas in the CMM,
- derive a profile based on an analysis of the satisfaction of the goals within the key process area, and
- present their results, to the appropriate audience, in terms of findings and a key process area profile.

4.2 Differences Between Software Process Assessments and Software Capability Evaluations

In spite of these similarities, the results of a software process assessment or software capability evaluation may differ, even on successive applications of the same method. One reason is that the scope of the assessment or evaluation may vary. First, the organization being investigated must be determined. For a large company, several different definitions for "organization" are possible. The scope may be based on common senior management, common geographical location, designation as a profit and loss center, common application domain, or other considerations. Second, even in what appears to be the same organization, the sample of projects selected may affect the scope. A division within a company may be assessed, with the team arriving at findings based on a division-wide scope. Later, a product line in that division may be evaluated, with that team arriving at its findings based on a much narrower scope. Comparison between the results without understanding the context is problematic.

Software process assessments and software capability evaluations differ in motivation, objective, outcome, and ownership of the results. These factors

lead to substantive differences in the dynamics of interviews, the scope of inquiry, the information gathered, and the formulation of the outcome. The assessment and evaluation methods are quite different when the detailed procedures employed are examined. Assessment training does not prepare a team to perform evaluations, or vice versa.

Software process assessments are performed in an open, collaborative environment. Their success depends on a commitment from both management and the professional staff to improve the organization. The objective is to surface problems and help managers and engineers improve their organization. While the questionnaire is valuable in focusing the assessment team on maturity level issues, the emphasis is on structured and unstructured interviews as tools for understanding the organization's software process. Aside from identifying the software process issues facing the organization, the buy-in to improvement, the organization-wide focus on process, and the motivation and enthusiasm in executing an action plan are the most valuable outcomes of an assessment.

Software capability evaluations, on the other hand, are performed in a more audit-oriented environment. The objective is tied to monetary considerations, since the team's recommendations will help select contractors or set award fees. The emphasis is on a documented audit trail that reveals the software process actually implemented by the organization.

This does not mean, however, that the results of software process assessments and software capability evaluations should not be comparable. Since both methods are CMM-based, the points of comparison and difference should be evident and explainable.

4.3 Other Uses of the CMM in Process Improvement

For software engineering process groups or others trying to improve their software process, the CMM has specific value in the areas of action planning, implementing action plans, and defining processes. During action planning, the members of the software engineering process group, equipped with knowledge of their software process issues and business environment, can compare their current practices against the goals of the key process areas in the CMM. The key practices should be examined in relation to corporate goals, management priorities, the level of performance of the practice, the value of implementing each practice to the organization, and the ability of the organization to implement a practice in light of its culture.

The software engineering process group must next determine which process improvements are needed, how to effect the change, and obtain the necessary buy-in. The CMM aids this activity by providing a starting point for discussion about process improvement and by helping to surface disparate assumptions about commonly accepted software engineering practices. In implementing the action plan, the CMM and the key practices can be used by the process groups to construct parts of the operational action plan and to define the software process.

5 Future Directions of the CMM

Achieving higher levels of software process maturity is incremental and requires a long-term commitment to continuous process improvement. Software organizations may take ten years or more to build the foundation for, and a culture oriented toward, continuous process improvement. Although a decade-long process improvement program is foreign to most U.S. companies, this level of effort is required to produce mature software organizations. This time frame is consistent with experience from other industries, such as the U.S. automotive industry, that have achieved significant gains in process maturity [Gabor90].

5.1 What the CMM Does Not Cover

The CMM is not a silver bullet [Brooks87] and does not address all of the issues that are important for successful projects. For example, the CMM does not currently address expertise in particular application domains, advocate specific software technologies, or suggest how to select, hire, motivate, and retain competent people. Although these issues are crucial to a project's success, some of these issues have been analyzed in other contexts [Curtis90]. They have not, however, been integrated into the CMM. The CMM was specifically developed to provide an orderly, disciplined framework within which to address software management and engineering process issues.

5.2 Near-Term Activities

Tutorials and courses on the CMM are being presented at major conferences and seminars throughout the United States to ensure that the software industry has adequate awareness of the CMM and its associated products. CMM-based tools (e.g., the maturity questionnaire), software process assessment training, and software capability evaluation training are being developed and/or revised to incorporate the CMM.

The near-term focus on CMM development activities will be oriented towards tailored versions of the CMM, such as a CMM for small projects and/or small organizations. CMM v1.1 is expressed in terms of the normative practices of large, government contracting organizations, and these practices must be tailored to the needs of organizations that differ from this template.

5.3 Long-Term Activities

During the next few years, the CMM will continue to undergo extensive testing through use in software process assessments and software capability evaluations. CMM-based products and training materials will be developed and revised as appropriate. The CMM is a living document that will be improved, but it is anticipated that CMM v1.1 will remain the baseline until at least 1996. This provides an appropriate and realistic balance between the needs for stability and for continued improvement.

For the next version of the CMM, Version 2, the SEI will turn its attention to improving the overall model. While all levels of the model may be revised, the emphasis will be on Levels 4 and 5. Currently the key process areas for Levels 2 and 3 have been the most completely defined. Since few organizations have been assessed to be at Levels 4 or 5 [Humphrey91a, Kitson92], less is known about the characteristics of such organizations. The practices for these two levels will be refined as the SEI works closely with organizations that are striving to understand and achieve Levels 4 and 5. The CMM may also become multi-dimensional to address technology and human resource issues.

The SEI is also working with the International Standards Organization (ISO) in its efforts to build international standards for software process assessment, improvement, and capability evaluation. The development of the ISO standards will influence CMM v2, even as the SEI's process work will influence the activities of the ISO.

5.4 Conclusion

Continuous improvement applies to the maturity model and practices, just as it does to the software process. The potential impact of changes to the CMM on the software community will be carefully considered, but the CMM, the maturity questionnaire, and the software process assessment and software capability evaluation methods will continue to evolve as experience is gained with improving the software process. The SEI intends to work closely with industry, government, and academia in continuing this evolution.

The CMM provides a conceptual structure for improving the management and development of software products in a disciplined and consistent way. It does not guarantee that software products will be successfully built or that all problems in software engineering will be adequately resolved. The CMM identifies practices for a mature software process and provides examples of the state-of-the-practice (and in some cases, the state-of-the-art), but it is not meant to be either exhaustive or dictatorial. The CMM identifies the characteristics of an effective software process, but the mature organization addresses all issues essential to a successful project, including people and technology, as well as process.

6 References

Brooks87 F.P. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering," *IEEE Computer*, Vol. 20, No. 4, April 1987, pp. 10-19.

Crosby79 P.B. Crosby, *Quality is Free*, McGraw-Hill, New York, NY, 1979.

Curtis90 B. Curtis, "Managing the Real Leverage in Software Productivity and Quality," *American Programmer*, Vol. 3, No. 7, July 1990, pp. 4-14.

Deming86 W. Edwards Deming, *Out of the Crisis*, MIT Center for Advanced Engineering Study, Cambridge, MA, 1986.

DoD87 *Report of the Defense Science Board Task Force on Military Software*, Office of the Under Secretary of Defense for Acquisition, Washington, D.C., September 1987.

Fagan86 M.E. Fagan, "Advances in Software Inspections," *IEEE Transactions on Software Engineering*, Vol. 12, No. 7, July, 1986, pp. 744-751.

Fowler90 P. Fowler and S. Rifkin, *Software Engineering Process Group Guide*, Software Engineering Institute, CMU/SEI-90-TR-24, ADA235784, September, 1990.

Freedman90 D.P. Freedman and G.M. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews, Third Edition*, Dorset House, New York, NY, 1990.

Gabor90 — A. Gabor, *The Man Who Discovered Quality*, Random House, New York, NY, 1990.

GAO-92-48 — *Embedded Computer Systems: Significant Software Problems on C-17 Must Be Addressed*, GAO/IMTEC-92-48, May 1992.

Humphrey87a — W.S. Humphrey, *Characterizing the Software Process: A Maturity Framework*, Software Engineering Institute, CMU/SEI-87-TR-11, ADA182895, June 1987.

Humphrey87b — W.S. Humphrey and W.L. Sweet, *A Method for Assessing the Software Engineering Capability of Contractors*, Software Engineering Institute, CMU/SEI-87-TR-23, ADA187320, September 1987.

Humphrey88 — W.S. Humphrey, "Characterizing the Software Process," *IEEE Software*, Vol. 5, No. 2, March, 1988, pp. 73-79.

Humphrey89 — W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, Reading, MA, 1989.

Humphrey91a — W.S. Humphrey, D.H. Kitson, and J. Gale, "A Comparison of U.S. and Japanese Software Process Maturity," *Proceedings of the 13th International Conference on Software Engineering*, Austin, TX, 13-17 May 1991, pp. 38-49.

Humphrey91b — W.S. Humphrey, "Process Fitness and Fidelity," *Proceedings of the Seventh International Software Process Workshop*, 16-18 October 1991.

IEEE-STD-610 — ANSI/IEEE Std 610.12-1990, "IEEE Standard Glossary of Software Engineering Terminology," February 1991.

Imai86 — M. Imai, *Kaizen: The Key to Japan's Competitive Success*, McGraw-Hill, New York, NY, 1986.

Juran88 — J.M. Juran, *Juran on Planning for Quality*, Macmillan, New York, NY, 1988.

Juran89 — J.M. Juran, *Juran on Leadership for Quality*, The Free Press, New York, NY, 1989.

Kitson92 — D.H. Kitson and S. Masters, *An Analysis of SEI Software Process Assessment Results: 1987-1991*, Software Engineering Institute, CMU/SEI-92-TR-24, July 1992.

Paulk91 — M.C. Paulk, B. Curtis, M.B. Chrissis, et al, *Capability Maturity Model for Software*, Software Engineering Institute, CMU/SEI-91-TR-24, ADA240603, August 1991.

Paulk93a — M.C. Paulk, B. Curtis, M.B. Chrissis, and Charles V. Weber, *Capability Maturity Model for Software, Version 1.1*, Software Engineering Institute, CMU/SEI-93-TR-24, February 1993.

Paulk93b — M.C. Paulk, C.V. Weber, S. Garcia, M.B. Chrissis, and M. Bush, *Key Practices of the Capability Maturity Model, Version 1.1*, Software Engineering Institute, CMU/SEI-93-TR-25, February 1993.

Radice85 — R.A. Radice, J.T. Harding, P.E. Munnis, and R.W. Phillips, "A Programming Process Study," *IBM Systems Journal*, Vol. 24, No.2, 1985.

Siegel90 — J.A.L. Siegel, et al., *National Software Capacity: Near-Term Study*, Software Engineering Institute, CMU/SEI-90-TR-12, ADA226694, May 1990.

Weber91 — C.V. Weber, M.C. Paulk, C.J. Wise, and J.V. Withey, *Key Practices of the Capability Maturity Model*, Software Engineering Institute, CMU/SEI-91-TR-25, ADA240604, August 1991.

Appendix A: Goals for Each Key Process Area

Goals for each key process area are listed by maturity level below.

A.1 The Key Process Areas for Level 2: Repeatable

Requirements Management

Goal 1 System requirements allocated to software are controlled to establish a baseline for software engineering and management use.

Goal 2 Software plans, products, and activities are kept consistent with the system requirements allocated to software.

Software Project Planning

Goal 1 Software estimates are documented for use in planning and tracking the software project.

Goal 2 Software project activities and commitments are planned and documented.

Goal 3 Affected groups and individuals agree to their commitments related to the software project.

Software Project Tracking and Oversight

Goal 1 Actual results and performances are tracked against the software plans.

Goal 2 Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans.

Goal 3 Changes to software commitments are agreed to by the affected groups and individuals.

Software Subcontract Management

Goal 1 The prime contractor selects qualified software subcontractors.

Goal 2 The prime contractor and the software subcontractor agree to their commitments to each other.

Goal 3 The prime contractor and the software subcontractor maintain ongoing communications.

Goal 4 The prime contractor tracks the software subcontractor's actual results and performance against its commitments.

Software Quality Assurance

Goal 1 Software quality assurance activities are planned.

Goal 2 Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively.

Goal 3 Affected groups and individuals are informed of software quality assurance activities and results.

Goal 4 Noncompliance issues that cannot be resolved within the software project are addressed by senior management.

Software Configuration Management

Goal 1 Software configuration management activities are planned.

Goal 2 Selected software work products are identified, controlled, and available.

Goal 3 Changes to identified software work products are controlled.

Goal 4 Affected groups and individuals are informed of the status and content of software baselines.

A.2 The Key Process Areas for Level 3: Defined

Organization Process Focus

Goal 1 Software process development and improvement activities are coordinated across the organization.

Goal 2 The strengths and weaknesses of the software processes used are identified relative to a process standard.

Goal 3 Organization-level process development and improvement activities are planned.

Organization Process Definition

Goal 1 A standard software process for the organization is developed and maintained.

Goal 2 Information related to the use of the organization's standard software process by the software projects is collected, reviewed, and made available.

Training Program

Goal 1 Training activities are planned.

Goal 2 Training for developing the skills and knowledge needed to perform software management and technical roles is provided.

Goal 3 Individuals in the software engineering group and software-related groups receive the training necessary to perform their roles.

Integrated Software Management

Goal 1 The project's defined software process is a tailored version of the organization's standard software process.

Goal 2 The project is planned and managed according to the project's defined software process.

Software Product Engineering

Goal 1 The software engineering tasks are defined, integrated, and consistently performed to produce the software.

Goal 2 Software work products are kept consistent with each other.

Intergroup Coordination

Goal 1 The customer's requirements are agreed to by all affected groups.

Goal 2 The commitments between the engineering groups are agreed to by the affected groups.

Goal 3 The engineering groups identify, track, and resolve intergroup issues.

Peer Reviews

Goal 1 Peer review activities are planned.

Goal 2 Defects in the software work products are identified and removed.

A.3 The Key Process Areas for Level 4: Managed

Quantitative Process Management

Goal 1 The quantitative process management activities are planned.

Goal 2 The process performance of the project's defined software process is controlled quantitatively.

Goal 3 The process capability of the organization's standard software process is known in quantitative terms.

Software Quality Management

Goal 1 The project's software quality management activities are planned.

Goal 2 Measurable goals for software product quality and their priorities are defined.

Goal 3 Actual progress toward achieving the quality goals for the software products is quantified and managed.

A.4 The Key Process Areas for Level 5: Optimizing

Defect Prevention

Goal 1 Defect prevention activities are planned.

Goal 2 Common causes of defects are sought out and identified.

Goal 3 Common causes of defects are prioritized and systematically eliminated.

Technology Change Management

Goal 1 Incorporation of technology changes are planned.

Goal 2 New technologies are evaluated to determine their effect on quality and productivity.

Goal 3 Appropriate new technologies are transferred into normal practice across the organization.

Process Change Management

Goal 1 Continuous process improvement is planned.

Goal 2 Participation in the organization's software process improvement activities is organization wide.

Goal 3 The organization's standard software process and the projects' defined software processes are improved continuously.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-93-TR-24	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESC-TR 93-177
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute 6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office
6c. ADDRESS (city, state, and zip code) Carnegie Mellon University Pittsburgh PA 15213	7b. ADDRESS (city, state, and zip code) ESC/AVS Hanscom Air Force Base, MA 01731
8a. NAME OF FUNDING/SPONSORING ORGANIZATION SEI Joint Program Office 8b. OFFICE SYMBOL (if applicable) ESC/AVS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003
8c. ADDRESS (city, state, and zip code)) Carnegie Mellon University Pittsburgh PA 15213	10. SOURCE OF FUNDING NOS PROGRAM ELEMENT NO: 63756E PROJECT NO: N/A TASK NO: N/A WORK UNIT NO: N/A
11. TITLE (Include Security Classification) Capability Maturity Model for Software, Version 1.1	
12. PERSONAL AUTHOR(S) Mark C. Paulk, Bill Curtis, and Mary Beth Chrissis	
13a. TYPE OF REPORT Final 13b. TIME COVERED FROM TO	14. DATE OF REPORT (year, month, day) February 1993 15. PAGE COUNT 63 pp.
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	capability maturity model software process capability software process performance

19. ABSTRACT (continue on reverse if necessary and identify by block number)

In November 1986, the Software Engineering Institute (SEI) with assistance from the Mitre began developing a process maturity framework that would assist organizations in improving their software process. This effort was initiated in response to a request to provide the federal government with a method for assessing the capability of their software contractors. In September 1987, the SEI released a brief description of the process maturity framework and a maturity questionnaire (CMU/SEI-87-TR-23). The SEI intended the maturity questionnaire to provide a simple tool for identifying areas where an organization's software process needed improvement. Unfortunately, the questionnaire was too often regarded as "the model" rather than as a vehicle for exploring process maturity issues.

After four years of experience with the software process maturity framework and the preliminary version of the maturity questionnaire, the SEI has evolved the software process maturity framework into a fully defined model.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ■ SAME AS RPT ☐ DTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution
22a. NAME OF RESPONSIBLE INDIVIDUAL Thomas R. Miller, Lt Col, USAF	22b. TELEPHONE NUMBER (include area code) (412) 268-7631 22c. OFFICE SYMBOL ESC/AVS (SEI)

ABSTRACT — continued from page one, block 19

This model will be used in a systematic, principled way to derive a maturity questionnaire. By fully elaborating the maturity framework, a model has emerged that provides organizations with more effective guidance for establishing process improvement programs than was offered by the maturity questionnaire. Using knowledge acquired from software process assessments and extensive feedback from both industry and government, an improved version of the process maturity framework has been produced called the Capability Maturity Model for Software (CMM). This paper is an introduction to the revised model.

SUPPLEMENTARY

INFORMATION

ERRATA. AD-A263403

Technical Report
CMU/SEI-93-TR-024
ESC-TR-93-177
February 1993

Capability Maturity Model[SM] for Software, Version 1.1

Mark C. Paulk
Bill Curtis
Mary Beth Chrissis
Charles V. Weber

Unlimited distribution subject to the copyright.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

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

Copyright © 1996 by Carnegie Mellon University.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

Requests for permission to reproduce this document or to prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTIBILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through Research Access, Inc., 800 Vinial Street, Pittsburgh, PA 15212.
Phone: 1-800-685-6510. FAX: (412) 321-2994. RAI also maintains a World Wide Web home page. The URL is http://www.rai.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145. Phone: (703) 274-7633.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.