

1991

Capability Maturity Model for Software

Mark Paulk
Carnegie Mellon University

Bill Curtis
Carnegie Mellon University, bcurtis@andrew.cmu.edu

Mary Beth C. Chrissis
Carnegie Mellon University, mb@sei.cmu.edu

Follow this and additional works at: <http://repository.cmu.edu/sei>

 Part of the [Software Engineering Commons](#)

This Technical Report is brought to you for free and open access by Research Showcase @ CMU. It has been accepted for inclusion in Software Engineering Institute by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

AD-A240 603



August 1991
CMU/SEI-91-TR-24
ESD-TR-91-24

2

Capability Maturity Model for Software



Mark C. Paulk
Bill Curtis
Mary Beth Chrissis

Edward L. Averill
Judy Bamberger
Timothy C. Kasse
Mike Konrad
Jeffrey R. Perdue
Charles V. Weber
James V. Withey

Approved for public release
Distribution unlimited

91-11242



Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This document was prepared for the


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

The ideas and findings in this document 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 document has been reviewed and is approved for publication.

FOR THE COMMANDER


Charles J. Ryan, Major, USAF
SEI Joint Program Office

The Software Engineering Institute is sponsored by the U.S. Department of Defense. This report was funded by the Department of Defense.

Copyright © 1991 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.

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



Carnegie Mellon University
Software Engineering Institute

September 12, 1991

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
Sr. Editor and Manager
SEI Information Management

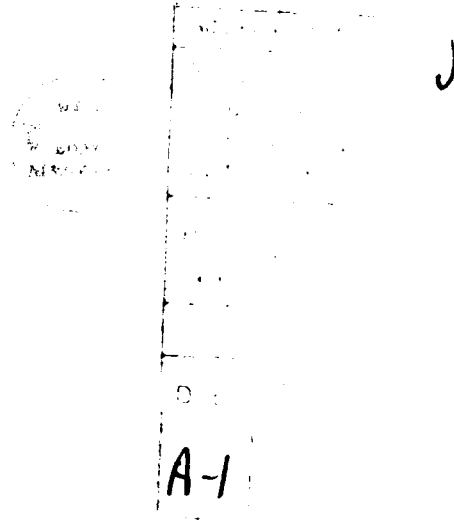


Table of Contents

Acknowledgements.....	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.....	11
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 Managed and Optimizing Levels	14
2.3 Visibility Into the Software Process	15
2.4 Process Capability and the Prediction of Performance.....	18
2.5 Skipping Maturity Levels.....	20
3 Operational Definition of the Capability Maturity Model.....	23
3.1 Internal Structure of the Maturity Levels	24
3.2 Maturity Levels	26
3.3 Key Process Areas.....	26
3.4 Key Practices.....	29
3.5 Key Indicators	32
3.6 Maturity Questionnaire.....	32

Table of Contents

4	Using the CMM.....	35
4.1	Software Process Assessment and Software Capability Evaluation Methods.....	36
4.2	Differences Between Software Process Assessments and Software Capability Evaluations.....	41
4.3	Other Uses of the CMM in Process Improvement.....	42
5	Future Directions of the CMM.....	45
5.1	What the CMM Does Not Cover.....	45
5.2	Near Term Activities.....	45
5.3	Long Term Activities.....	46
5.4	Conclusion.....	46
6	References.....	49
	Appendix A: Goals for Each Key Process Area.....	53
A.1	The Key Process Areas for Level 2 - Repeatable.....	53
A.2	The Key Process Areas for Level 3 - Defined.....	55
A.3	The Key Process Areas for Level 4 - Managed.....	57
A.4	The Key Process Areas for Level 5 - Optimizing.....	58

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.....	15
Figure 2.3	A Management View of Visibility into the Software Process at Each Maturity Level	16
Figure 2.4	The Capability as Indicated by Maturity Level	19
Figure 3.1	The CMM Structure.....	25
Figure 3.2	Building the CMM Structure: An Example of a Maturity Level.....	26
Figure 3.3	The Key Process Areas by Maturity Level	28
Figure 3.4	Building the CMM Structure: An Example of a Key Process Area.....	29
Figure 3.5	Building the CMM Structure: An Example of a Key Practice	31
Figure 3.6	Building the CMM Structure: An Example of a Question in the Maturity Questionnaire	34
Figure 4.1	Common Steps in Software Process Assessments and Software Capability Evaluations	37
Figure 4.2	The Key Process Area Profile Template.....	40

List of Figures

Acknowledgements

The description of the Capability Maturity Model for Software was produced by a dedicated group of people who spent many hours discussing the model and its features and then trying to capture it in this paper. 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 in his efforts and took over in his absence.

At various stages, several people made contributions to this paper. They include Mary Merrill, George Pandelios, Anita Carleton, Marty Carlson, Richard Kauffold, Steve Masters, Jim Over, and Jane Siegel. These people attended meetings that helped the group formulate and come to consensus on the meaning of software process maturity.

Joe Besselman and Betty Deimel provided comments and suggestions as reviewers. We appreciate the administrative help from Dorothy Josephson, Debbie Punjack, Carolyn Tady, Marcia Theoret, Andy Tsounos and Todd Bowman; and the editorial assistance from Mary Beth Chrissis and Bill Pollak. Renne Dutkowski from the American Institutes for Research provided suggestions for the design of the document.

Acknowledgements

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 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 (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 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. 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.

To the Reader

What is the Purpose of This Paper?

This paper provides a technical overview of the Capability Maturity Model for Software and reflects the most current version. Specifically, this paper describes the process maturity framework, the structural additions 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 the earlier model and questionnaire.

The SEI's goal in working with industry and government to refine and expand the model is to encourage software organizations to focus on the CMM rather than on the maturity questionnaire. The SEI has developed a series of new products to encourage this focus. This paper, in combination with the "Key Practices of the Capability Maturity Model," is intended to help software organizations use the CMM as a guide to improve the maturity of their software process.

Who Should Read This Paper?

This paper presents an introduction to the CMM and all 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.

How is This Paper Organized?

This paper has five chapters.

<i>Chapter 1</i>	Defines the concepts necessary to understand the CMM and the motivation and purpose behind it.
<i>Chapter 2</i>	Describes the five levels of the CMM and the principles that underlie them.
<i>Chapter 3</i>	Describes how the CMM is structured into key process areas, refined into key practices, and used to construct the maturity questionnaire.
<i>Chapter 4</i>	Provides a high-level overview of how the CMM provides guidance for software process assessments, software capability evaluations, and process improvement programs.
<i>Chapter 5</i>	Concludes by providing a description of future directions for the CMM and its related products.

To the Reader

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 products listed below help you understand and use the CMM. All of these products have been systematically derived from the model.

"Key Practices of the Capability Maturity Model"	Identifies the software practices for each level of the CMM.
<i>Maturity Questionnaire</i>	Used by software process assessment and software capability evaluation teams to identify strengths and weaknesses in an organization's software process.
<i>Respondent Questionnaire</i>	Used to collect information on the individual completing the maturity questionnaire.
<i>Project Questionnaire</i>	Used to collect information about the project for which the maturity questionnaire is being completed.

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

How Do You Receive More Information?

This paper and the "Key Practices of the Capability Maturity Model" will be available directly from the Defense Technical Information Center (DTIC) or the National Technical Information Service (NTIS) by the end of 1991. Several versions of the maturity questionnaire will enter a pilot test phase beginning in Fall 1991. When a final version has emerged from this phase, it will be available from DTIC and NTIS. These documents can be obtained by writing to:

Defense Technical Information Center
Attn: Cameron Station
Alexandria, VA 22304-6145

National Technical Information Service
U.S. Department of Commerce
Springfield, VA 22161

Other requests regarding the CMM and its associated products can be made to:

CMM Info
Software Process Program
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7700
Internet: cmm-info@sei.cmu.edu

To the Reader

1 The Process Maturity Framework

After two decades of unfulfilled promises about productivity and quality gains from applying new software methods and technologies, industry and government organizations are realizing that their fundamental problem is the inability to manage the software process. 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, 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 at 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

The Process Maturity Framework

project. Even when 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. 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 in order 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 effectively. The processes mandated are 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 product 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.

1.2 Fundamental Concepts Underlying Process Maturity

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 a mature, disciplined software process. 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.

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." Thus, 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, user manuals, etc.). 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 from following a software process. The software process capability of an organization provides one means of predicting the most likely outcomes that are expected from the next software project the organization undertakes.

Software process performance represents the actual results achieved from following a software process. Thus, software process performance focuses

The Process Maturity Framework

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 performance of the project may not reflect the full process capability of the organization. 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 performance to fall short of the organization's full process capability.

Software process maturity implies a potential for growth in capability. Maturity indicates both the richness of an organization's software process and the consistency with which it is applied in projects throughout the organization. The CMM is based on the premise that maturity is an indicator of capability. 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 applying 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 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 often 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. These stages must be ordered 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 organizations embarking on the journey of 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 process for developing and maintaining software and how to evolve toward a culture of software engineering 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, organizations can steadily improve their organization-wide software process to enable continuous and lasting gains in software process capability.

The staged structure of the CMM is based on product quality principles that have existed for the last sixty years. In the 1930s Walter Shewart, a physicist at AT&T Bell Laboratories, 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 into a maturity framework that establishes the project management and engineering foundation during the initial stages and during the more advanced stages of maturity to quantitatively control the process.

The maturity framework into which these quality principles have been adapted was first inspired by Philip Crosby of ITT 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 (Radice85) working under the direction of Watts Humphrey at IBM. Humphrey brought this maturity framework to the Software Engineering

The Process Maturity Framework

Institute in 1986, revised it to add the concept of maturity levels, and developed the foundation for its current use throughout the software industry.

Early versions of Humphrey's maturity framework were 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 process. Two methods, software process assessment and software capability evaluation, were developed in 1987 to help organizations use previous versions of the CMM in appraising software process maturity.¹ During 1990-1991 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.

¹Software process assessments help software organizations identify their most important improvement needs. Software capability evaluations help acquisition agencies identify the risks associated with awarding business to contractors, as well as manage on-going contracts.

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.

A *maturity level* is a well-defined evolutionary plateau on the path toward becoming a mature software organization. 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.

The Five Levels of Software Process Maturity

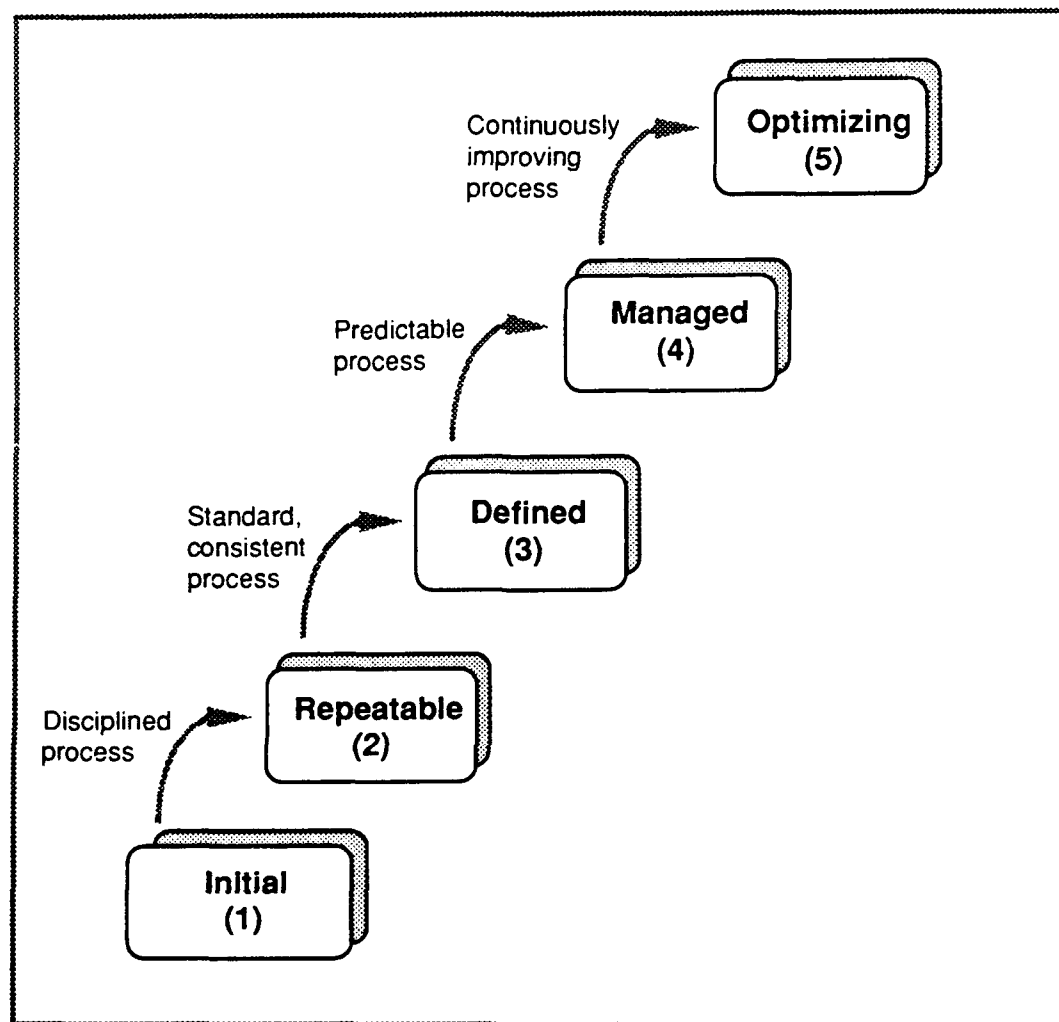


Figure 2.1 The Five Levels of Software Process Maturity

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.

The Five Levels of Software Process Maturity

- 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 an organization-wide software process. All projects use a documented and approved version of the organization's 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 using detailed measures.
- 5) *Optimizing* Continuous process improvement is enabled by quantitative feedback from the process and from testing 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

The Five Levels of Software Process Maturity

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 sound management practices are lacking within an organization, the benefits of software engineering practices are undermined by ineffective planning and reaction-driven commitment systems.

Projects typically abandon planned procedures and revert to coding and testing during a crisis. Success depends entirely on having an exceptional manager and a seasoned and effective software team. Occasionally, unusually 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. A strong engineering process cannot overcome the instability created by the absence of sound management practices.

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

The Five Levels of Software Process Maturity

2.1.2 Level 2 - The Repeatable Level

At the Repeatable level, software project management policies and procedures are established by the organization. Planning and managing new projects is based on experience with similar projects. An objective in achieving Level 2 is to stabilize software project management processes, which allows organizations to repeat successful practices developed on earlier projects.

Projects in Level 2 organizations have installed basic software management controls. Realistic project commitments are established from results observed on previous projects. Project managers track software costs, schedules, and functionality; and problems in meeting commitments are identified when they arise. Software requirements and the artifacts developed to satisfy them are baselined, and their integrity is controlled. Project standards are defined, and the organization ensures they are faithfully followed. Project teams work with their customers, and their subcontractors if any, to establish a stable, managed, working environment.

The process capability of Level 2 organizations can be summarized as stable for planning and tracking the software project, because a disciplined management process provides a project environment for repeating earlier successes. The 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

The Five Levels of Software Process Maturity

engineering and software management processes, and they are integrated into a coherent whole. Processes established at the Defined level are used by both management and staff and are changed as appropriate to help them perform more effectively. The organization exploits effective software engineering practices when standardizing software processes for the organization. There is a permanent organizational focus on the software process; a software engineering process group (SEPG) facilitates process definition and improvement efforts (Fowler90). An organization-wide training program is implemented to ensure that all practitioners and managers have the knowledge and skills required to carry out their tasks.

Projects use the organization-wide standard software process for developing and maintaining software as a basis for creating their own defined software process that encompasses the unique characteristics of the project. Each project uses a peer review process to enhance product quality. Because the software process is well defined, management has good visibility into technical progress on all projects. Management and engineering activities are coherently integrated on each project.

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

2.1.4 Level 4 - The Managed Level

At the Managed level, the organization sets quantitative quality goals for software products. Productivity and quality are measured for important software process activities across all projects in the organization. An organization-wide process database is used to collect and analyze the data available from a carefully defined process. Software processes have been instrumented with well-defined and consistent measures at Level 4. The

The Five Levels of Software Process Maturity

measures establish the quantitative foundation for evaluating project processes and products.

Projects achieve control over their products and processes by narrowing the variation in their performance to within acceptable quantitative boundaries. Meaningful variations in performance can be distinguished from random variation (noise), particularly within established product lines. In order to reduce the process variation due to constant shifts among new application domains, there is a strategic business plan describing which product lines to pursue. The risks involved in moving up the learning curve of a new application domain are known and carefully managed.

The process capability of Level 4 organizations can be summarized as measured and operating within measurable limits. This level of process capability allows an organization to predict process and product quality trends within the quantitative bounds of these limits. When these limits are violated, 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 organization is focused on continuous process improvement. The organization has the means to identify weak process elements and strengthen them, with the goal of preventing the occurrence of defects. Statistical evidence is available on process effectiveness and is used in performing cost benefit analyses on new technologies. Innovations that exploit the best software engineering practices are identified.

Project teams analyze process performance to determine the causes of defects. Software processes are evaluated to prevent known types of defects from recurring, and lessons learned are disseminated to other projects.

The Five Levels of Software Process Maturity

Level 5 organizations are continuously striving to raise the upper bound of their process capability. Improvement occurs both by incremental advancements in the existing process and by innovations using new technologies and methods.

2.2 Understanding the Managed and Optimizing Levels

Maturity Levels 4 and 5 are unknown territory for the software industry in 1991. There are a few examples of Level 4 and 5 software projects, but none for Level 4 and 5 software organizations (Humphrey89b). Unfortunately, these projects are too few to draw general conclusions about 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. This Juran Trilogy Diagram illustrates the primary responsibilities of process management. 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 stable overall. The second responsibility, and the focus of Level 5, is continuous process improvement: the software process is changed to move the zone of quality control; that is, to establish a new baseline for performance and reduce chronic waste. The lessons learned in improving such a process are applied in planning future processes.

It is anticipated that organizations reaching the highest maturity levels of the CMM would have a process that is capable of producing extremely

The Five Levels of Software Process Maturity

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.

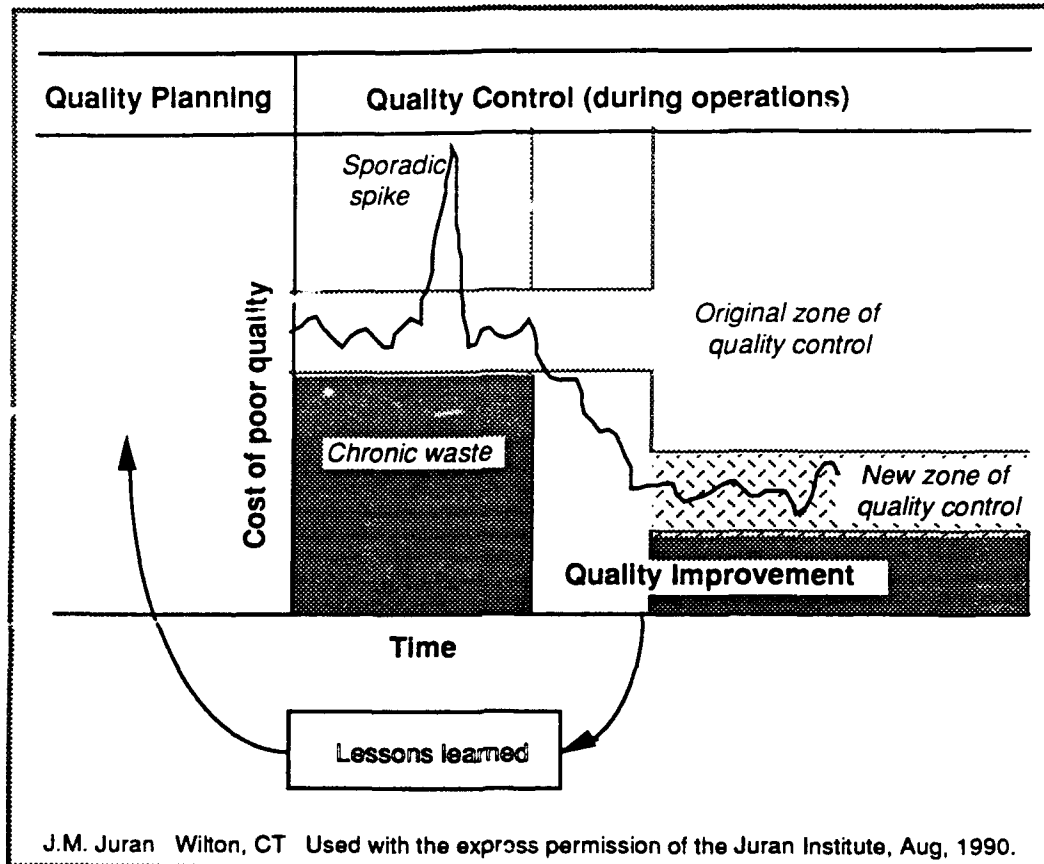


Figure 2.2 The Juran Trilogy Diagram: Quality Planning, Quality Control, and Quality Improvement

2.3 Visibility Into the Software Process

Software engineers have 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 from their experience with a

The Five Levels of Software Process Maturity

particular subsystem. Those outside the project without first-hand exposure, such as senior managers, lack visibility into project 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 software processes.

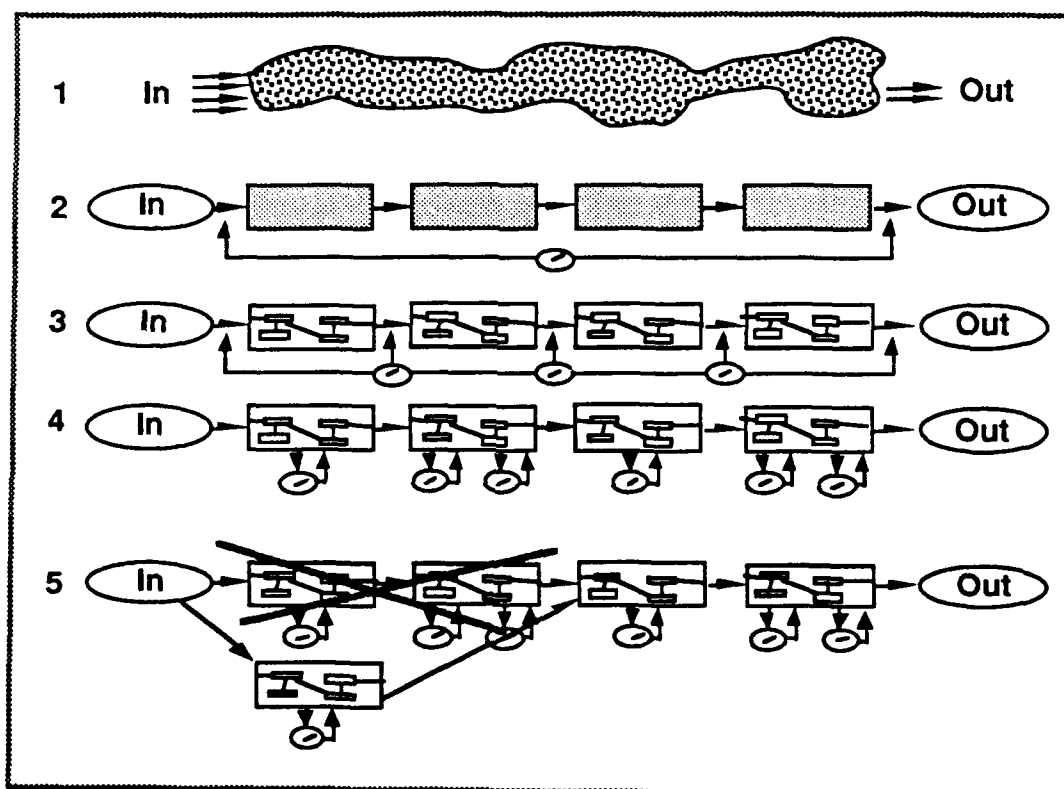


Figure 2.3 A Management View of Visibility Into the Software Process at Each Maturity Level

At Level 1, the software process is an amorphous entity, and visibility into project processes is difficult. Since the staging of activities is ambiguous, managers have an extremely difficult time establishing the status of project progress and activities. Requirements flow into the software process in an uncontrolled manner, and a product results. Software development is

The Five Levels of Software Process Maturity

frequently viewed as black magic, especially by managers who are unfamiliar with software.

At Level 2, the customer requirements and product artifacts are controlled, and basic project management practices have been established. These management controls allow visibility into the project only on defined occasions. The process of building the product can be viewed as a succession of black boxes that allows management visibility only 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.

At Level 3, the internal structure of the boxes - the subprocesses in the software process defined by the project - are visible. The internal structure represents the way the organization-wide 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. 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 product are continually tried, in a controlled manner, to improve productivity and quality. 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 by the example 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 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 mass under the curve lies to the right of the target line.)

The Five Levels of Software Process Maturity

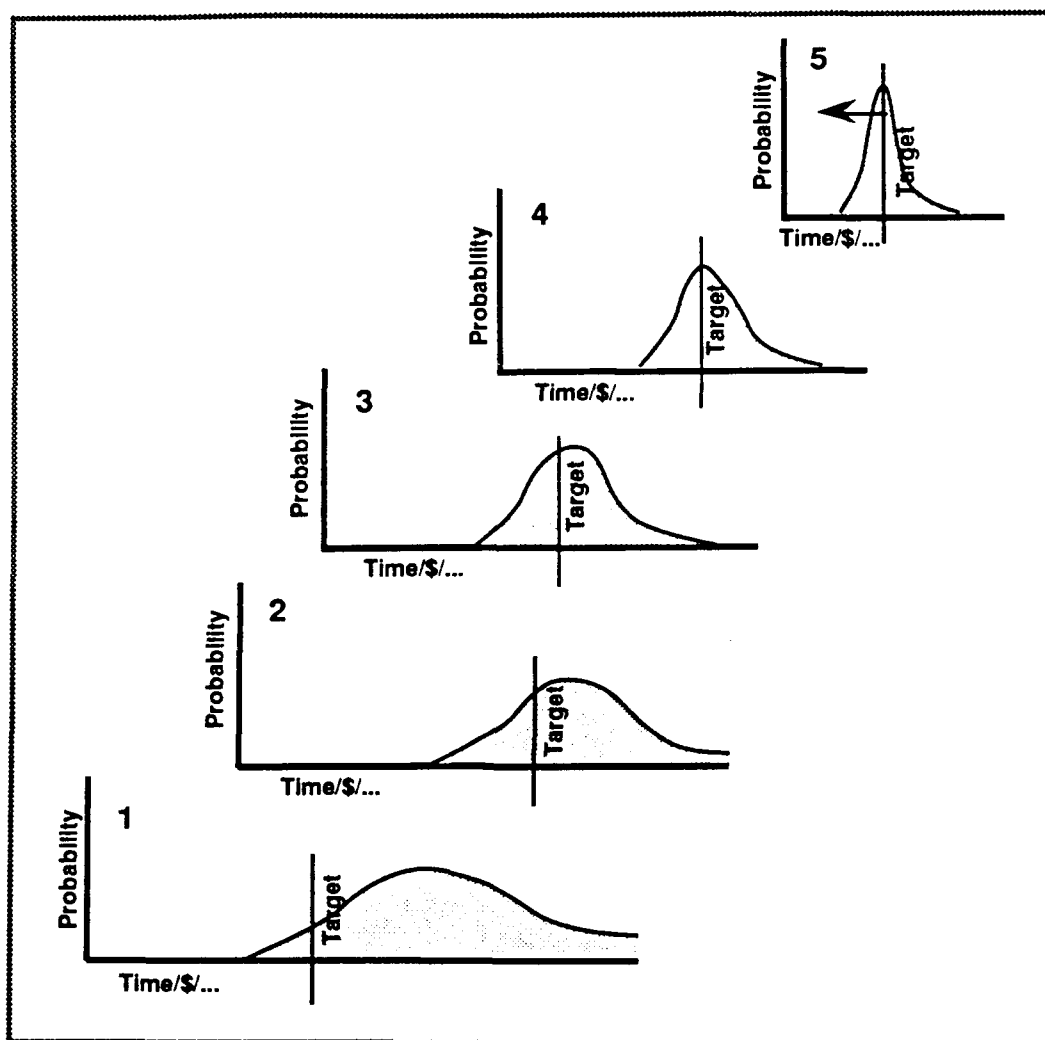


Figure 2.4 The Capability as Indicated by Maturity Level

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 of dates. 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

The Five Levels of Software Process Maturity

cost, schedule, functionality, and quality. (This is illustrated in Figure 2.4 by how much of the mass 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 project results represented in Figure 2.4 assume that software project outcomes become more predictable as noise, often in the form of rework, is removed from the software process. 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. Continuous process improvement allows Level 5 organizations to continually improve on their targeted performance.

2.5 Skipping Maturity Levels

Because of their impatience for results, senior management occasionally attempts to reach Level 5 without progressing through Levels 2, 3, and 4 in sequence. This is counterproductive, however, because each level forms a necessary foundation from which to construct 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

The Five Levels of Software Process Maturity

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 has not established a defined process (Level 3) and is trying to implement a managed process (Level 4) is usually unsuccessful because without defined processes, there is no common basis for interpreting measurements. While data can be collected, few of the metrics have significant meaning across projects, and they do not increase the understanding of the software process. It is difficult to identify meaningful metrics in the absence of defined processes because of the variation in the processes being measured.

An organization that has not established a managed process (Level 4) and is trying to implement an optimizing process (Level 5) is likely to fail. Without controlling the process within statistically narrow boundaries (small variations in process measures), there is too much noise in the data to objectively determine 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 Five Levels of Software Process Maturity

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. In order to use the CMM to guide improvement programs, it must be defined to describe improvement actions in terms of software activities. This operational elaboration of the CMM must be designed to support the many ways it will be used in improvement programs. There are at least four uses of the CMM that must be supported.

- ❑ Assessments teams will use the CMM to identify improvements needed in the organization.
- ❑ Evaluation teams will use the CMM to identify the risks of selecting among different contractors for awarding business and as a tool to monitor contracts.
- ❑ Managers will use the CMM to understand the activities necessary to implement an improvement program across their organization.
- ❑ Process improvement groups will use the CMM as a guide to help them define and improve the software process in their organization.

The diversity in use of the CMM requires that it be decomposed in sufficient detail that actual process recommendations can be derived from the structure of the maturity levels. This decomposition also makes it possible to determine which among these recommended processes are the most salient indicators of 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 bridges from abstract summaries of each level down to their operational definition of the items composing the maturity questionnaire as shown in Figure 3.1. Thus, a maturity level is composed of several key process areas. Each key process area consists of numerous key practices that, when addressed collectively, accomplish the goals of the key process area. Some of the key practices are selected as key indicators of whether the goals of a key process area are accomplished. These key practices will be selected to become questions in the maturity questionnaire.

Operational Definition of the Capability Maturity Model

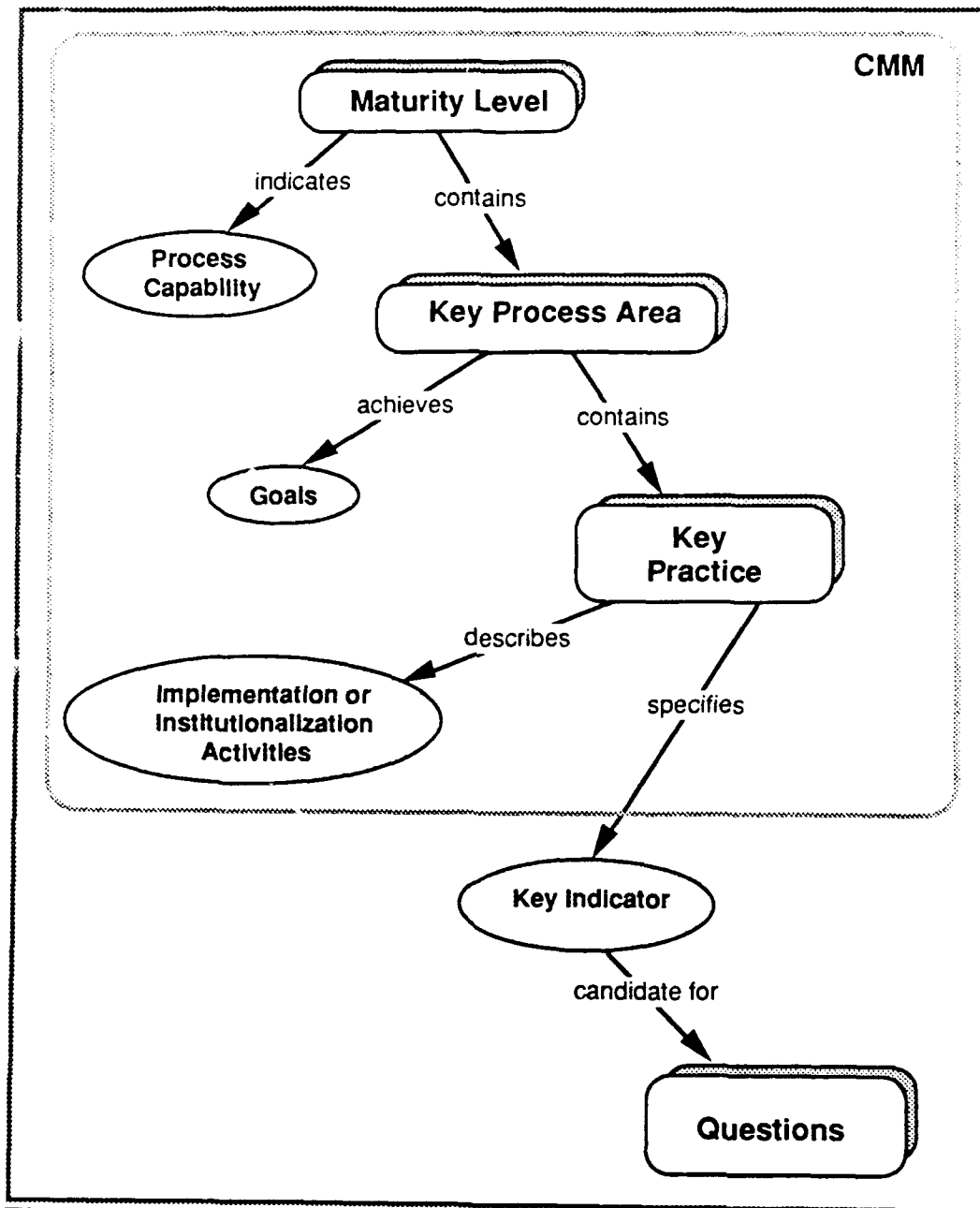


Figure 3.1 The CMM Structure

3.2 Maturity Levels

Each maturity level indicates a level of process capability, as illustrated in Figure 3.2. For instance, at Level 2 (the Repeatable level) the process capability of an organization has been elevated by establishing a disciplined process under sound project management control.

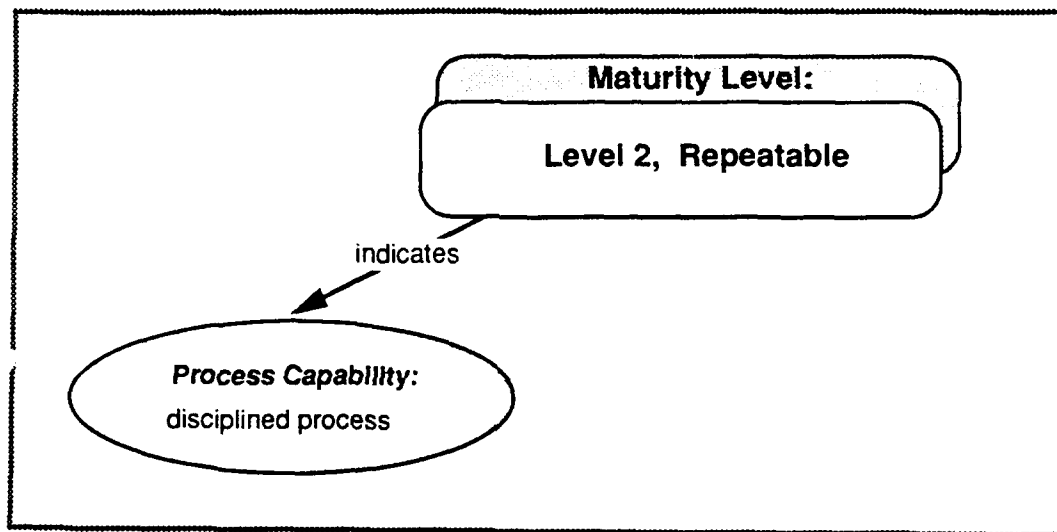


Figure 3.2 Building the CMM Structure: An Example of a Maturity Level

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 their software process. Key process areas identify the issues that must be addressed in order to achieve a maturity level.

Operational Definition of the Capability Maturity Model

Although other issues affect process performance, the key process areas were identified because of their effectiveness in improving an organization's software capability. They may be considered the requirements for achieving a maturity level.

Figure 3.3 displays the key process areas for each maturity level. In order to achieve a maturity level, the goals of each key process area at that level must be satisfied.

The key process areas have been defined to reside at a single maturity level as shown in Figure 3.3. The specific practices to be executed in each key process area will evolve in content 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 additional project data available at Levels 3, 4, and 5. For example, at Level 3, the Integrated Software Management key process area was defined to capture the evolution from managing a project according to a plan to managing a project using a defined software process.

Each key process area represents a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing process capability. An example of a goal to be achieved by the Software Project Planning key process area is presented in Figure 3.4. 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 in order for the organization to satisfy that key process area.² When the goals of a key process area have been accomplished on a continuing basis, the organization can be said to have institutionalized the process capability characterized by the key process area.

² For a listing of the goals for each key process area, refer to Appendix A.

Operational Definition of the Capability Maturity Model

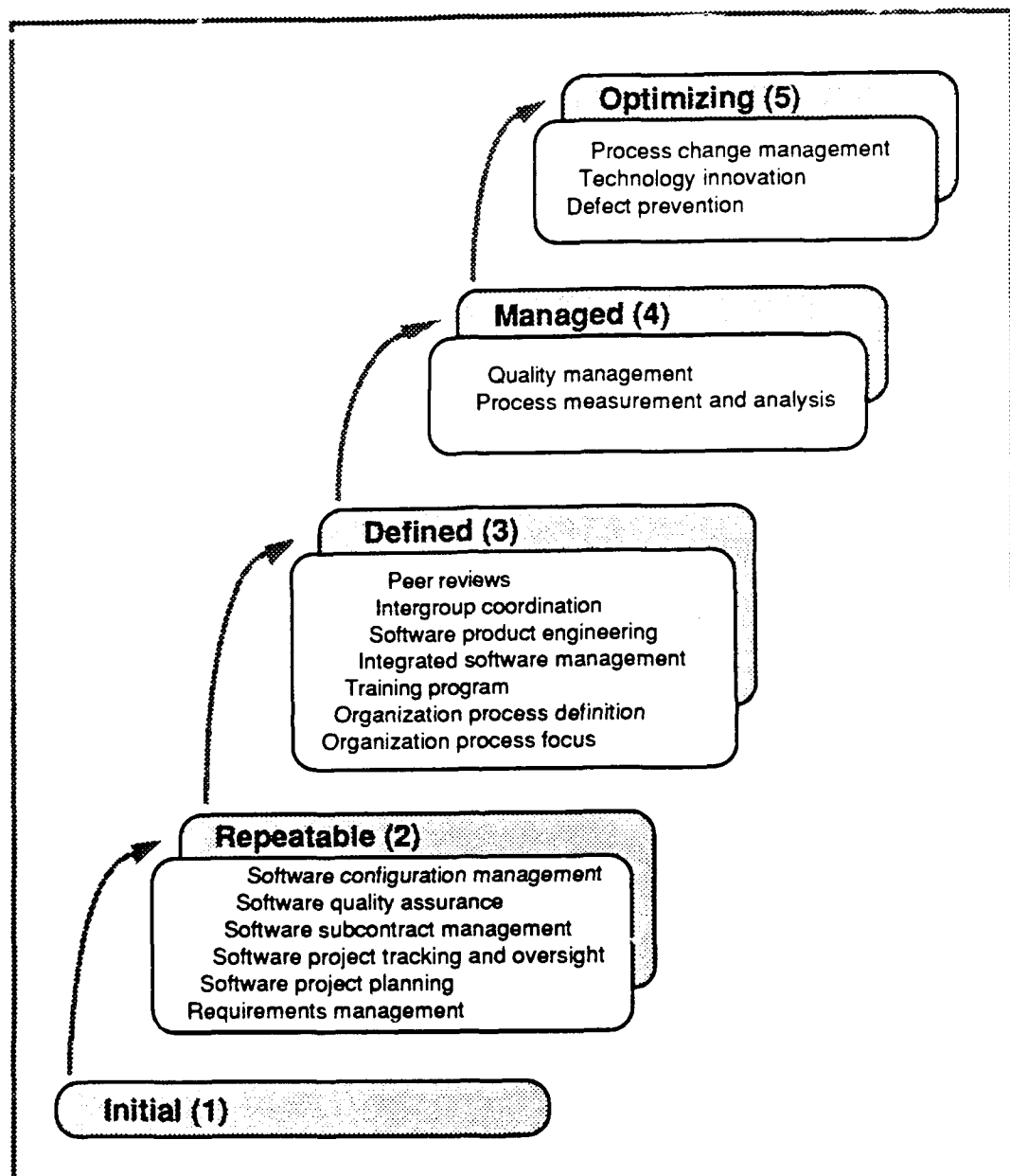


Figure 3.3 The Key Process Areas by Maturity Level

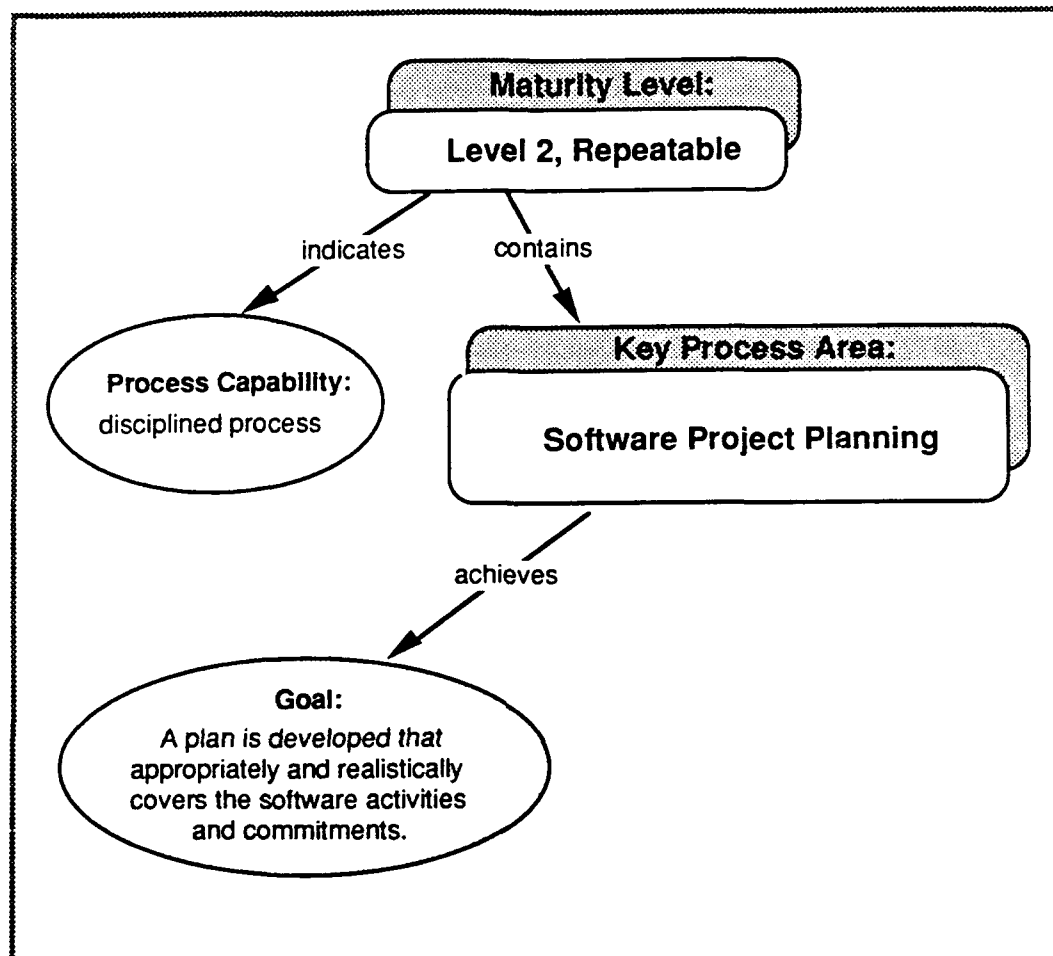


Figure 3.4 Building the CMM Structure: An Example of a Key Process Area

3.4 Key Practices

Each key process area is defined by the key practices that contribute to satisfying its goals. The *key practices* are the policies, procedures, and activities that most contribute to the effective institutionalization and implementation of the key process area. At the highest level of abstraction, the goals themselves represent the key practices of a key process area.

Operational Definition of the Capability Maturity Model

However, in order to provide guidance for improving an organization's performance in a key process area, more detailed practices must be presented. Therefore, each key process area was broken down into the common features listed in italics below.

- ❑ Practices such as establishing policies and procedures demonstrate an organization's *commitment to perform* in the key process area.
- ❑ Practices such as providing training, sufficient resources, and appropriate tools support an organization's *ability to perform* in the key process area.
- ❑ Practices that are required to make effective product building decisions describe the *activities performed* in the key process area.
- ❑ Practices such as collecting measures and tracking data allow an organization to *monitor the implementation* of the activities constituting a key process area.
- ❑ Practices such as management reviews allow an organization to *verify the implementation* of the activities constituting a key process area.

The practices in the common feature, activities performed, describe what must be addressed in a project to improve the process capability. The other practices, taken as a whole, form the basis by which an organization can institutionalize and reduce the variance in the gains achieved by addressing the practices included in the activities performed common feature.

Each key practice consists of a single sentence, often followed by a more detailed description, which may include examples and elaboration. Figure 3.5 illustrates a top-level key practice for the Software Project Planning key process area.

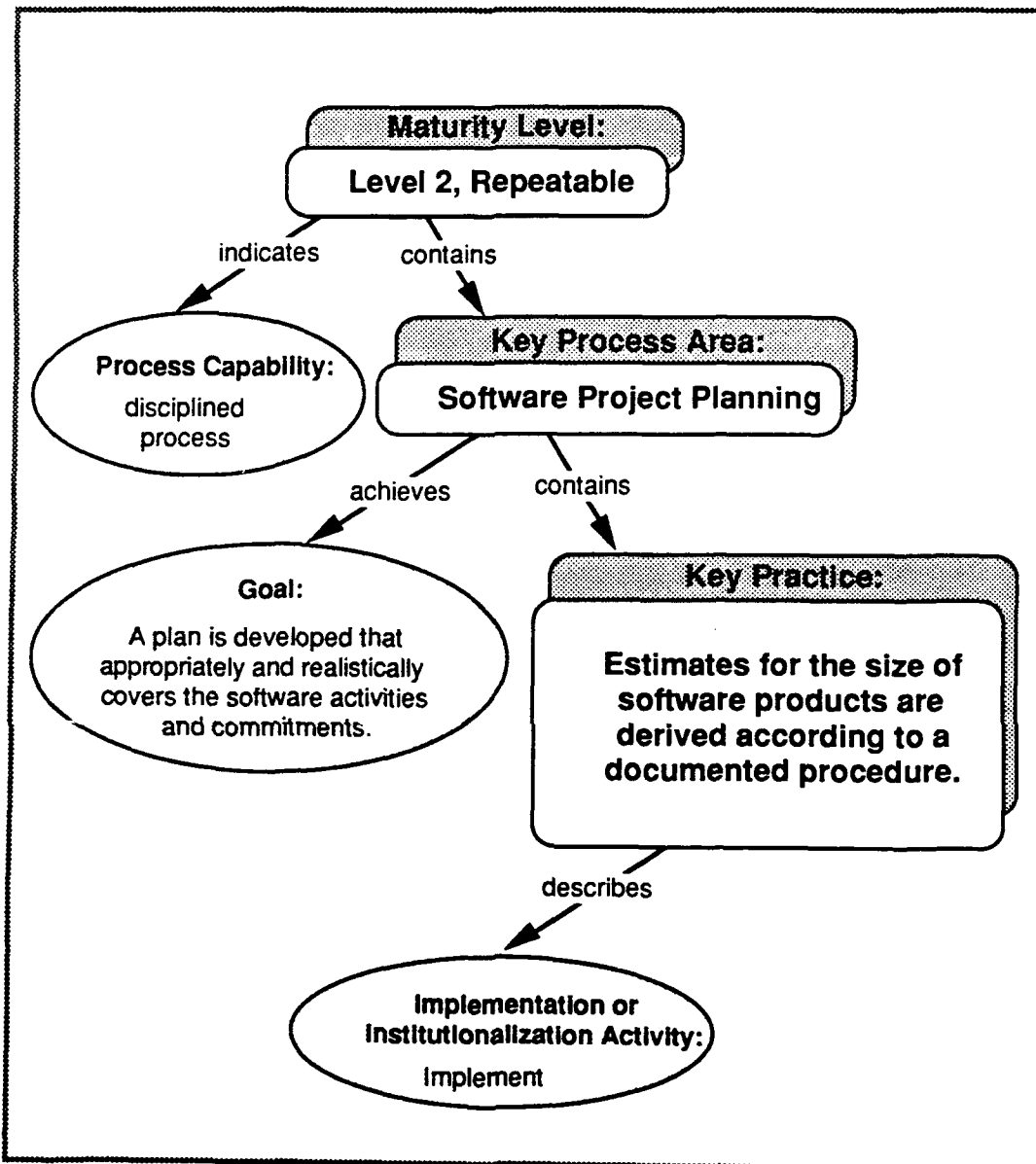


Figure 3.5 Building the CMM Structure: An Example of a Key Practice

In order to ensure consistent accomplishment of the goal of developing plans that appropriately and realistically cover the software activities and commitments, the organization must establish a documented procedure for

Operational Definition of the Capability Maturity Model

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 estimates. These criteria guide appraisal of whether a reasonable size estimating procedure is followed; simply documenting that "George does our size estimation" is inadequate.

The key practices describe what is to be accomplished, but *they should not be interpreted as mandating how* the key practices should be performed. An important consideration is whether an alternative process accomplishes the goals of the key process area. The key practices should be interpreted rationally in the project environment to judge whether the goals of the key process area are effectively, although perhaps differently, achieved. The key practices for all key process areas are contained in the "Key Practices of the Capability Maturity Model," along with guidance on their interpretation.

3.5 Key Indicators

Key indicators are those key practices, or components of a key practice, that offer the greatest insight into whether the goals of a key process area have been satisfied. Key indicators were determined through a lengthy process involving workshop recommendations, industry comments, software process assessment and software capability evaluation findings, statistical evaluation of the preliminary questionnaire, and the professional judgement of the SEI staff.

3.6 Maturity Questionnaire

The maturity questionnaire presents a set of yes/no questions about the software process that sample the practices in each key process area of the

Operational Definition of the Capability Maturity Model

CMM. Key indicators provide the basis for developing the maturity questionnaire. The maturity questionnaire is used by an assessment or evaluation team to initiate the investigation of an organization's software process capability.

An example of a question developed from the CMM is presented in Figure 3.6. The key practice indicating that estimates for the size of software products are derived according to a documented procedure has been selected as a key indicator of the Software Project Planning key process area. One question corresponding to this key indicator is: "Do you use a documented procedure to estimate software size (e.g., lines of code, function points, etc.)?" This question is included in the pool of candidate questions that will be evaluated in pilot testing of the maturity questionnaire to ensure that the wording is not confusing and that the question is a good discriminator of whether the goals of the Software Project Planning key process area have been satisfied.

Other questions might be developed to investigate an organization's size estimating procedure at a more detailed level than described in the statement of the key practice. For example, one element of a size estimating procedure is that historical sizing data are used. Accordingly, the question, "Do you use historical size data to help derive software size estimates?" may prove to be a good indicator of whether an effective software planning process for the project is in place and is being used. In fact, there are many questions that could be asked to determine whether an effective software planning process has been established. The goal in selecting candidate questions for the maturity questionnaire from among the key indicators is to identify those that best reveal the true state of an organization's process.

Operational Definition of the Capability Maturity Model

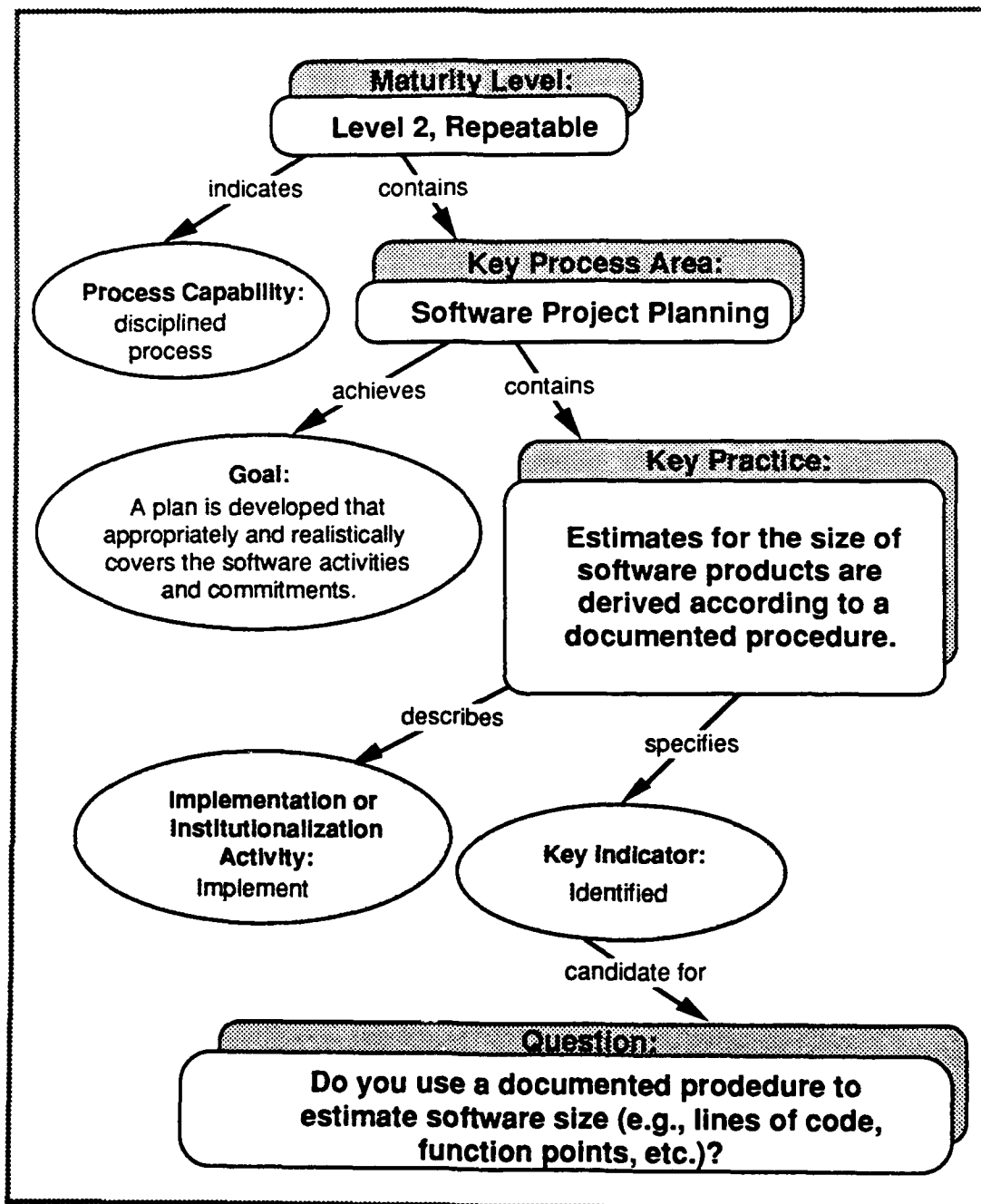


Figure 3.6 Building the CMM Structure: An Example of a Question in the Maturity Questionnaire

4 Using the CMM

The CMM establishes a set of public criteria describing the characteristics of mature software organizations. These criteria can be used by organizations to improve their process 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 by organizations to help identify the status of their software process and to identify a prioritized list of areas to address for process improvement.
- ❑ *Software capability evaluations* are used by acquisition agencies to identify contractors who are qualified to build high-quality software and also to monitor contract performance.

The information presented here is not sufficient for readers to conduct either an assessment or a capability evaluation. Anyone wishing to apply the CMM through these methods should receive training, since an undisciplined application of the model may serve an organization poorly.

The CMM is a common foundation for both software process assessments and software capability evaluations. The purpose of the methods are quite different and there are significant differences in the specific methods used. However, both are based on the model and the products derived from it. The revisions made to the 1987 software process maturity framework that are reflected in the CMM will not cause structural changes in the software process assessment or software capability evaluation methods. Rather, these changes will enable the methods to be applied more reliably and lead to more effective results.

4.1 Software Process Assessment and Software Capability Evaluation Methods

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

Software capability evaluations are conducted by acquisition agencies and are focused on identifying contractors who represent the lowest risk 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 for contract monitoring purposes, with the intent of identifying 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.

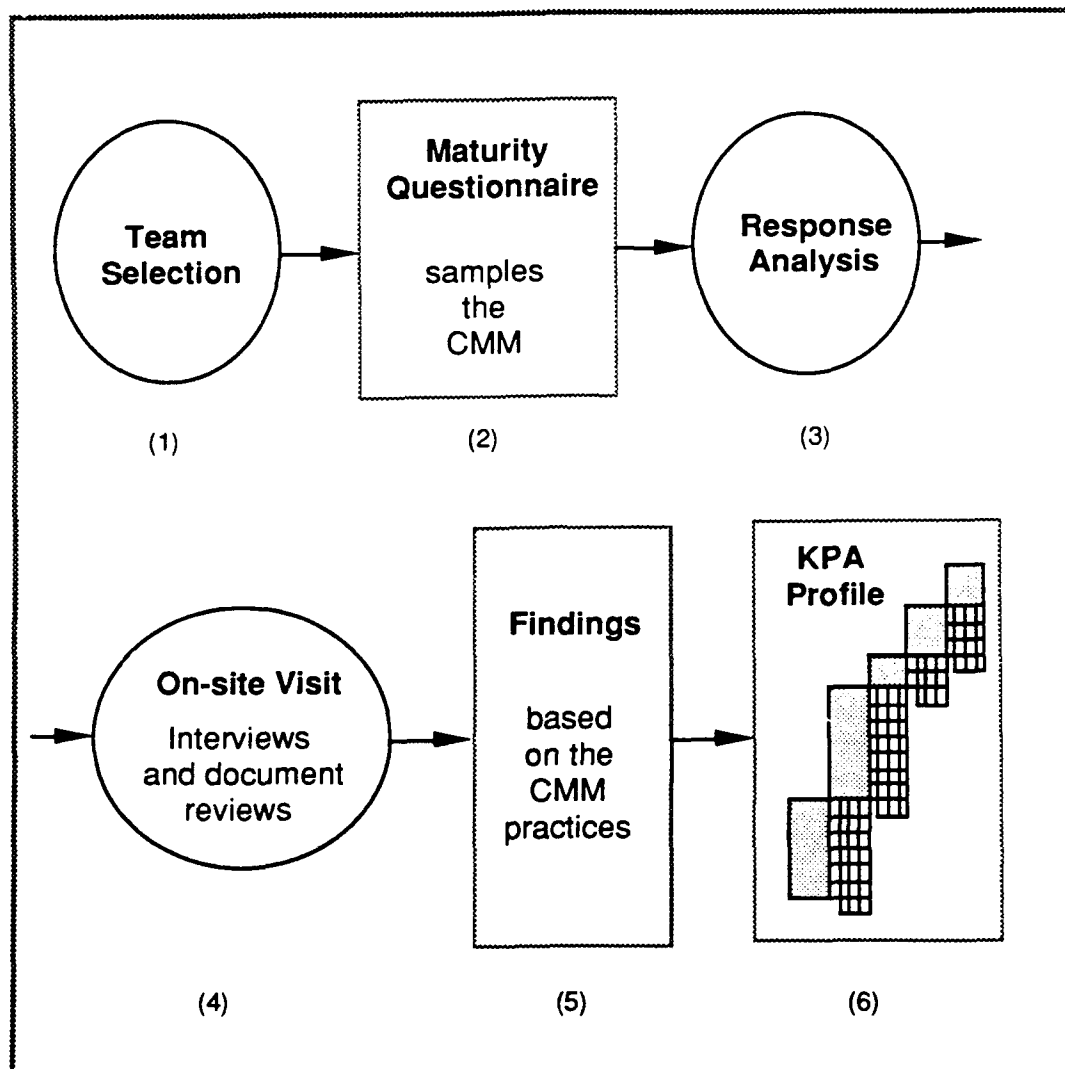


Figure 4.1 Common Steps in Software Process Assessments and Software Capability Evaluations

The first step in both of these methods is to select a team. This team should attend a multi-day training course that presents the fundamental concepts of the CMM as well as the specifics of the assessment or evaluation method.

Using the CMM

The second step is to have representatives from the site to be assessed or evaluated complete the maturity questionnaire. These representatives are typically project or software managers from various projects. 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. Since each question in the maturity questionnaire was derived directly from the key practices in the CMM, the areas to be investigated correspond to the CMM key process areas. In order to develop an initial maturity profile, the key process areas for each maturity level are listed and the degree of satisfaction is appraised for each based on the initial responses to the maturity questionnaire. This profile focuses the attention of the assessment or capability evaluation team on the maturity level and key process areas deserving investigation.

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 CMM provides guidance and examples of practices from mature software organizations; the CMM is not a prescription for the "one right way" to a mature software process. The team applies professional judgement in deciding whether the site's implementation of the key process areas satisfies the relevant key process area goals. When there are clear differences between the key practices in the CMM and the site's practices, the team must document their rationale for appraising that key process area.

At the end of the on-site period, the assessment or evaluation 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 contracting agency.

The findings are grounded in key practices, which is a shift in emphasis from the questions in the maturity questionnaire to the key practices in the model underlying the questions. Responses to questions on the maturity questionnaire provide a point of departure for investigating whether a project has developed realistic processes for satisfying key process area goals. If the goals of a key process area have not been satisfied, an assessment or capability evaluation team may report a finding against that area, regardless of how many questions were answered yes.

Also, 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 is completed (see Figure 4.2). Each key process area is rated as not satisfied (NS), partially satisfied (PS), or fully satisfied (FS). Each key process area within a maturity level must be fully satisfied for the organization to be designated as functioning at that maturity level. A key process area can be fully satisfied and still have associated findings, provided the findings do not identify major problems that inhibit achieving any goals of the key process areas.

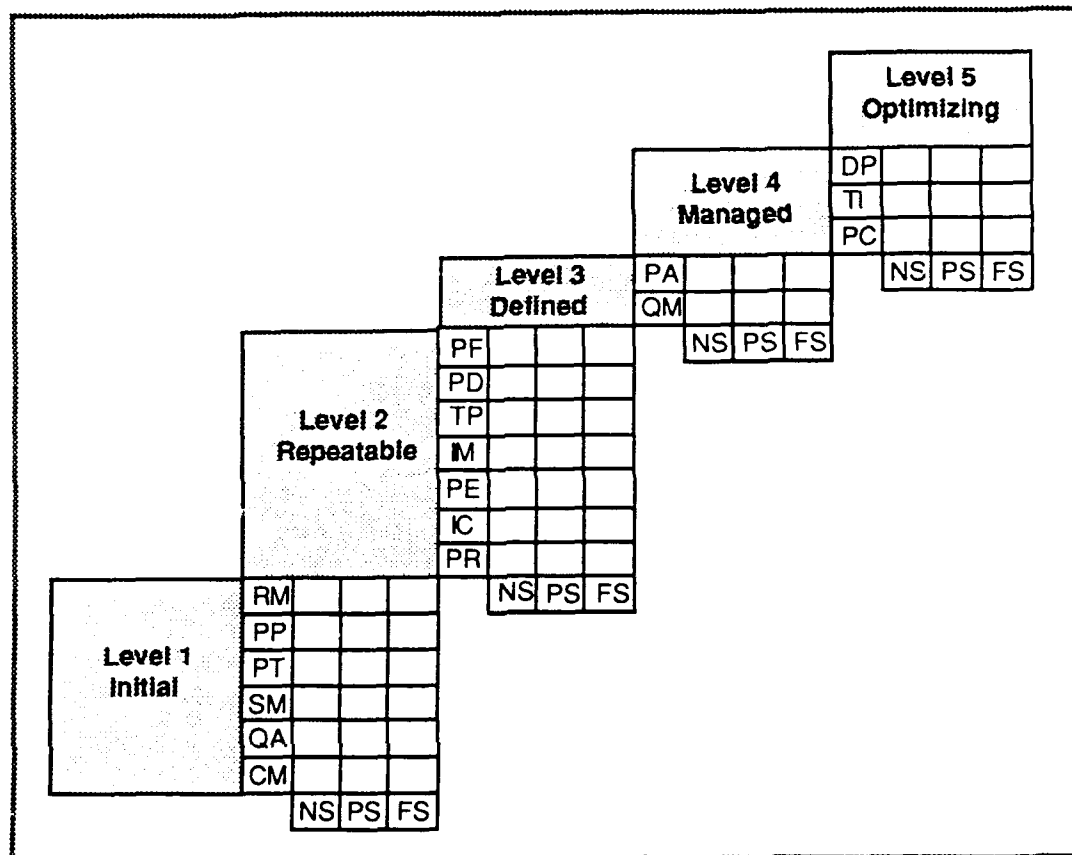


Figure 4.2 The Key Process Area Profile Template

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
- ☐ present their results, to the appropriate audience, in terms of a key process area profile and findings

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 be quite different, even on successive applications of the same method. One reason is that the scope of the assessment or evaluation may vary. First, the site being investigated must be determined. For a large company, several different definitions for a site are possible. These definitions 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 site, the scope may differ depending on the selection of projects. A division within a company may be assessed, and the team arrives at findings based on a division-wide scope. Later, a product line may be evaluated, and that team arrives at its findings based on a much narrower scope that is based on the product. When the process has not been institutionalized, as in the case of Level 1 organizations, there may be dramatic differences between projects in the same organization. Comparison between the results then becomes an apples-versus-oranges problem.

Software process assessments and software capability evaluations differ in motivation, objective, outcome, and ownership of the results. These factors lead to substantive differences in interview dynamics, scope of inquiry, information being gathered, and 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. The interview and document review processes are used to validate the responses to the maturity questionnaire, and therefore several types of interviews are required during the site visit.

4.3 Other Uses of the CMM in Process Improvement

For software engineering process groups (SEPGs) or others trying to improve their software process, the CMM has specific value in the areas of action planning, action plan implementation, and process definition. During action planning, the software engineering process group members, equipped with knowledge of their software process issues, can compare their current practices against the goals in 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 this practice to the organization, and the ability of the organization to implement this practice in light of its culture.

The software engineering process group must next determine which key practices to address 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 project success, some of these issues have been analyzed in other contexts (e.g., Curtis90). However, they have not been integrated into the CMM. The CMM was specifically developed to provide an orderly, disciplined framework within which to address management and engineering process issues.

5.2 Near Term Activities

In the near term, the CMM will be extensively tested through use in software process assessments and software capability evaluations. Tutorials will be presented at major conferences and seminars throughout the United States to ensure that the software industry has ample opportunity for contact and awareness with this new CMM and its associated components.

Future Directions of the CMM

SEI software process assessment and software capability evaluation training courses will be revised to incorporate the new features of the CMM.

At least two cycles of course revision and field testing are anticipated during 1991 and 1992. During this training and field testing period many of the CMM components may be refined. Users of the CMM can anticipate that during this initial testing period, questions will change the most frequently, followed by the key practices, the key process areas, and then the model itself.

5.3 Long Term Activities

Following the initial testing and revision of the CMM, the SEI will turn its attention to improving the overall model itself. 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 no organizations have been assessed to be at levels 4 or 5 (Humphrey89b), 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 who are striving to understand and achieve Levels 4 and 5. The CMM may become multi-dimensional to address technology and human resource issues.

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

Future Directions of the CMM

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. While the maturity questionnaire samples key indicators of an effective software process, the CMM identifies the characteristics of an effective software process, the mature organization addresses all issues essential to a successful project - including people and technology - as well as process.

Future Directions of the CMM

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.
- 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, September, 1990.
- Gabor90 A. Gabor, *The Man Who Discovered Quality*, Random House, New York, NY, 1990.
- Humphrey87a W.S. Humphrey, "Characterizing the Software Process: A Maturity Framework," Software Engineering Institute, CMU/SEI-87-TR-11, DTIC Number ADA182895, June 1987.

References

- 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, DTIC Number ADA187320, September 1987.
- Humphrey88 W.S. Humphrey, "Characterizing the Software Process," IEEE Software, Vol. 5, No. 2, March, 1988, pp. 73-79.
- Humphrey89a W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, Reading, MA, 1989.
- Humphrey89b W.S. Humphrey, D.H. Kitson, and T. Kasse, "The State of Software Engineering Practice: A Preliminary Report," Software Engineering Institute, CMU/SEI-89-TR-01, DTIC Number ADA206573, February, 1989.
- 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.
- 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.

References

- Siegel90 J.A.L. Siegel, et al., "National Software Capacity: Near-Term Study," Software Engineering Institute, CMU/SEI-90-TR-12, DTIC Number 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, August 1991.

References

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 - The system requirements allocated to software provide a clearly-stated, verifiable, and testable foundation for software engineering and software management. The allocated requirements define the scope of the software effort. The allocated requirements and changes to the allocated requirements are incorporated into the software plans, products, and activities in an orderly manner.

Software Project Planning - A plan is developed that appropriately and realistically covers the software activities and commitments. All affected individuals and groups understand the software estimates and plans and commit to support them. The software estimates and plans are documented for use in tracking the software activities and commitments.

Software Project Tracking and Oversight - Actual results and performance of the software project are tracked against documented and approved plans. Corrective actions are taken when the actual results and performance of the software project deviate significantly from plans. Changes to software commitments are understood and agreed to by all affected groups and individuals.

Goals for Each Key Process Area

Software Subcontract Management - The prime contractor selects qualified subcontractors. The software standards procedures and product requirements for the subcontract comply with the prime contractor's commitments. Commitments between the prime contractor and subcontractor are understood and agreed to by both parties. The prime contractor tracks the subcontractor's actual results and performance against the commitments.

Software Quality Assurance - Compliance of the software product with applicable standards, procedures, and product requirements is independently confirmed. When there are compliance problems, management is aware of them. Senior management addresses non-compliance issues.

Software Configuration Management - Controlled and stable baselines are established for planning, managing, and building the system. The integrity of the system's configuration is controlled over time. The status and content of the software baseline are known.

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

Organizational Process Focus - Current strengths and weaknesses of the organization's software process are understood and plans are established to systematically address the weaknesses. A group is established with appropriate knowledge, skills, and resources to define a standard software process for the organization. The organization provides the resources and support needed to record and analyze the use of the organization's standard software process in order to maintain and improve it.

Organizational Process Definition - A standard software process definition for the organization is defined and maintained as a basis for stabilizing and improving the performance of software projects. Specifications of common software processes and documented process experiences from past and current projects are collected and available.

Training Program - The staff and managers have the skills and knowledge to perform their jobs. The staff and managers effectively use, or are prepared to use, the capabilities and features of the existing and planned work environment. The staff and managers are provided with opportunities to improve their professional skills.

Integrated Software Management - The planning and managing of each software project is based on the organization's standard software process. Technical and management data from past and current projects are available and used to effectively and efficiently estimate, plan, track, and replan the software projects.

Goals for Each Key Process Area

Software Product Engineering - Software product and process issues are properly addressed in the system requirements and system design. The software engineering activities are well-defined, integrated, and used consistently to produce a software system. State-of-the-practice software engineering tools and methods are used, as appropriate, to build and maintain the software system. Software engineering products that are consistent with each other and appropriate for building and maintaining the software system are systematically developed.

Intergroup Coordination - The project's technical goals and objectives are understood and agreed to by its staff and managers. The responsibilities assigned to each of the project groups, and the working interfaces between these groups, are known to all groups. The project groups are appropriately involved in intergroup activities and in identifying, tracking, and addressing intergroup issues. The project groups work as a team.

Peer Reviews - Product defects are identified and fixed early in the life cycle. Appropriate product improvements are identified and implemented early in the life cycle. The staff becomes more effective through a better understanding of their work products and knowledge of errors that can be prevented. A rigorous group process for reviewing and evaluating product quality is established and used.

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

Process Measurement and Analysis - The organization's standard software process is stable and under statistical quality control. The relationship between product quality, productivity, and product development cycle time is understood in quantitative terms. Special causes of process variation (i.e., variations attributable to specific applications of the process and not inherent in the process) are identified and controlled. Process performance measures are used to quantitatively manage the software project.

Quality Management - Measurable goals and priorities for product quality are established and maintained for each software project through interaction with the customer, end users, and project groups. Measurable goals for process quality are established for all groups involved in the software process. The software plans, design, and process are adjusted to bring forecasted process and product quality in line with the goals. Process measurements are used to quantitatively manage the software project.

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

Defect Prevention - Sources of product defects that are inherent or repeatedly occur in the application of the software processes are identified and eliminated.

Technology Innovation - The organization has a software process and technology capability to allow it to develop or capitalize on the best available technologies in the industry. Selection and transfer of new technology into the organization is orderly and thorough. Technology innovations are tied to quality and productivity improvements of the organization's standard software process.

Process Change Management - The organization's staff and managers are actively involved in setting quantitative, measurable improvement goals and in improving the software process. The organization's standard software process and the projects' defined software processes continually improve. The organization's staff and managers are able to use the evolving software processes and their supporting tools and methods properly and effectively.