

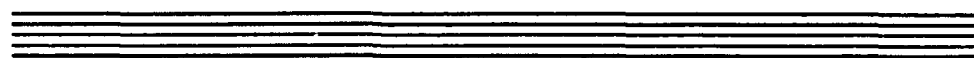
2

August 1991
CMU/SEI-91-TR-25
ESD-TR-91-25

AD-A240 604



Key Practices of the Capability Maturity Model



Charles V. Weber
Mark C. Paulk
Cynthia J. Wise
James V. Withey

Edited by:
Mary Beth Chrissis
Suzanne D. Couturiaux
Ginny Redish

Approved for public release
Distribution unlimited

F19628-85-C-0003

91-11243



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

Approved For	
NPS 0001 J	
Date	
By	
Date	
Approved For	
Date	Approved For
A-1	

Table of Contents

Acknowledgements.....	vii

Overview

1. Introducing the Key Practices Document	O-1
1.1 To the Reader.....	O-1
1.2 How Does this Document Relate to Other Documents?.....	O-3
1.3 How Is this Document Organized?.....	O-4
1.4 How Should this Document be Used?.....	O-5
2. Introducing the Capability Maturity Model.....	O-7
2.1 What Is the Capability Maturity Model?	O-7
2.2 Where Does the CMM Come From?.....	O-7
2.3 How Is the Model Structured?.....	O-8
2.4 What Do the Maturity Levels Mean?.....	O-11
2.4.1 Level 1 - The Initial Level.....	O-12
2.4.2 Level 2 - The Repeatable Level.....	O-13
2.4.3 Level 3 - The Defined Level	O-14
2.4.4 Level 4 - The Managed Level.....	O-15
2.4.5 Level 5 - The Optimizing Level.....	O-16
2.5 What Are Key Process Areas?.....	O-17
2.6 What Are Goals?.....	O-19
2.7 What Are the Key Practices?	O-19
2.8 What Are the Common Features?	O-20
3. Interpreting the Key Practices.....	O-23
3.1 How Should the Key Practices Be Used?	O-23
3.2 What Conventions Are Used in the Key Practices?.....	O-23
3.3 The Goals of Each Key Process Area Are the Essential Criteria.....	O-24
3.4 The CMM Does Not Imply a Particular Life-Cycle Model	O-25
3.5 The CMM Does Not Imply a Specific Software Technology	O-25
3.6 The CMM Does Not Require a Specific Set of Documents.....	O-25
3.7 The CMM Does Not Espouse Particular Roles or Organization Structure.....	O-26
3.8 The Example Organization Includes These Roles.....	O-28
3.9 The Example Organization Includes These Groups.....	O-30
3.10 The Actual Assignment of Roles May Vary	O-32

Table of Contents

3.11 The Composition of Groups May Vary.....	O-33
3.12 Independence of Responsibilities May Be a Concern On Small Projects	O-33
3.13 The CMM Allows Flexibility	O-34
3.14 The Organization's and Projects' Processes Must Have a Common Basis	O-35
4. Using the Key Practice Pages.....	O-41
Level 2 Key Practices	
Requirements Management.....	L2-1
Software Project Planning.....	L2-11
Software Project Tracking and Oversight.....	L2-29
Software Subcontract Management.....	L2-45
Software Quality Assurance	L2-61
Software Configuration Management.....	L2-71
Level 3 Key Practices	
Organization Process Focus	L3-1
Organization Process Definition.....	L3-11
Training Program	L3-23
Integrated Software Management.....	L3-33
Software Product Engineering	L3-57
Intergroup Coordination.....	L3-79
Peer Reviews	L3-89
Level 4 Key Practices	
Process Measurement and Analysis	L4-1
Quality Management	L4-15
Level 5 Key Practices	
Defect Prevention	L5-1
Technology Innovation.....	L5-17
Process Change Management.....	L5-31

Table of Contents

Appendices

Appendix A: References	A-1
Appendix B: Glossary of Terms	A-3
Appendix C: Abridged Version of the Key Practices	A-23
Appendix D: Change History	A-61

Change Request Form

Table of Contents

List of Figures

Figure 2.1	The Structure of the Capability Maturity Model.....	O-9
Figure 2.2	The Five Levels of Software Process Maturity	O-12
Figure 2.3	The Key Process Areas by Maturity Level.....	O-18
Figure 3.1	Conceptual Software Organization Used in the CMM.....	O-27
Figure 3.2	Conceptual Software Process Structure Used in the CMM...	O-36
Figure 4.1	Example of Key Practice Statements	O-43

List of Figures

Acknowledgements

The SEI would like to thank the many people who were involved in the development of this document. Over the last three years, the SEI has been revising the Capability Maturity Model for Software based on the knowledge acquired from software process assessments, software capability evaluations, and feedback from both industry and government. This document was produced as part of the revision effort.

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

This document has taken many drafts to evolve into the current product. We would like to thank the individuals who have worked many hours developing ideas and formulating the practices for this model. In particular, we acknowledge the contribution of Charlie Weber who led the effort, as well as the other authors Mark Paulk, Cynthia Wise, and Jim Withey.

Many other individuals have helped with their comments, criticisms, and suggestions. They include Jeff Perdue, Lionel Deimel, Judy Bamberger, Jim Rozum, Tim Kasse, Ed Averill, Mary Beth Chrissis, Mike Konrad, Anita Carleton, Bob Park, Ron Higuera, Mike Rissman, Julia Gale, Peter Feiler, George Pandelios, and Bill Curtis.

Special thanks go to the members of the CMM Working Group who contributed their time and effort to reviewing the drafts and providing insightful comments and recommendations. We would like to thank especially those who took time to participate in the March 1990 CMM Workshop and those who took time to meet with us on other occasions to provide comments and recommendations. This effort could not have been accomplished without these practitioners lending their expertise to refine the model.

Acknowledgements

We would also like to thank the members of the Questionnaire Advisory Board (Constance Ahara, Bill Curtis, Harry Carl, Conrad Czaplicki, Watts Humphrey, Martin Owens, Jerry Pixton, Ronald Willis, and Jim Withey) who helped guide us in our efforts. In addition to providing technical insights, they helped focus our effort and worked with us to evaluate and plan our actions to address the many comments received from industry and government reviewers.

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

1 Introducing the Key Practices Document

1.1 To the Reader

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

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

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

Introducing the key practices document

The CMM can be used for:

- ☐ software process assessments, in which an assessment team identifies improvements needed in the organization,
- ☐ software capability evaluations, in which an evaluation team identifies the risks of selecting among different bidders for awarding contracts and monitoring contracts, and
- ☐ software process improvement, in which an organization plans, defines, and implements changes to its software process.

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

This document can be used in several ways:

- ☐ by anyone wanting to understand the key practices that are part of an 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 effectively develop software,
- ☐ 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 in preparing teams to perform software process assessments or software capability evaluations.

1.2 How Does this Document Relate to Other Documents?

Two earlier documents provided the initial foundation for the CMM. They are:

- ❑ the paper, "Characterizing the Software Process" (Humphrey88), and
- ❑ the book, *Managing the Software Process* (Humphrey89).

To understand and use the CMM, you need two documents:

- ❑ the paper, "Capability Maturity Model for Software" (Paulk91), and
- ❑ this document, "Key Practices of the Capability Maturity Model."

In "Capability Maturity Model for Software," you will find an introduction to the model, including a brief history and rationale, an explanation of how organizations can use the maturity model to improve their processes, as well as descriptions of the five maturity levels of the CMM and descriptions of the specific key process areas that are covered in the maturity model.

In this document, "Key Practices of the Capability Maturity Model," you will find the key practices that correspond to the key process areas at each maturity level of the CMM and information on how to interpret the key practices.

Introducing the key practices document

The maturity questionnaire is derived from the maturity model and the key practices. The questionnaire can be used in both software process assessments and software capability evaluations. As an assessment tool, the questionnaire, together with the key practices, can help an organization understand the level of maturity of its software process and, therefore, understand what is needed to improve its software process. As an evaluation tool, the questionnaire can help acquisition personnel recognize whether an organization has a software process that is able to perform successfully on the contract.

The SEI has developed and is developing other products to support both software process assessments and software capability evaluations, including training courses, handbooks, and site visit guides.

1.3 How Is this Document Organized?

This first chapter is one of four that gives an overview of the CMM and of this document. In the next three chapters of the overview, you will find:

- ❑ an introduction to the CMM and its constituent parts,
- ❑ a description of how to use and interpret the key practices, and
- ❑ a description of how to use the format of the key practices.

Following the overview, the key practices for each level of the CMM are defined. The key practices begin with Level 2. For those who want to get a quick sense of the key practices, without the rigor that is needed in applying them, an abridgement of the key practices is provided in Appendix C.

In the appendices, you will find a list of the references cited in this document, a glossary of terms used in this document, an abridgement of the key practices, and the change history for this document.

1.4 How Should this Document be Used?

If you are not familiar with the CMM, you should first read the paper, "Capability Maturity Model for Software" (Paulk91). Then, read all four chapters in this overview before trying to use the key practice pages.

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

Introducing the key practices document

2 Introducing the Capability Maturity Model

2.1 What Is the Capability Maturity Model?

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

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

The CMM establishes standards against which it is possible to judge, in a repeatable way, the maturity of an organization's software process. These standards can also be used by an organization to plan improvements to its software process.

2.2 Where Does the CMM Come From?

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

Introducing the Capability Maturity Model

Earlier versions of the model and the questionnaire have been used in assessing numerous software organizations. We have gained additional knowledge about effective software processes from these software process assessments. We have also gained additional knowledge from meetings and workshops with industry and government representatives and from studying nonsoftware organizations. Using this additional knowledge, we have revised the maturity model and elaborated the corresponding practices.

2.3 How Is the Model Structured?

Figure 2.1 shows the structure of the CMM. The components of the CMM include:

<i>Maturity levels</i>	A maturity level is a well-defined evolutionary plateau on the path toward becoming a mature software organization. The five maturity levels provide the top-level structure of the CMM.
<i>Process capability</i>	Software process capability describes the range of results expected 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.

Introducing the Capability Maturity Model

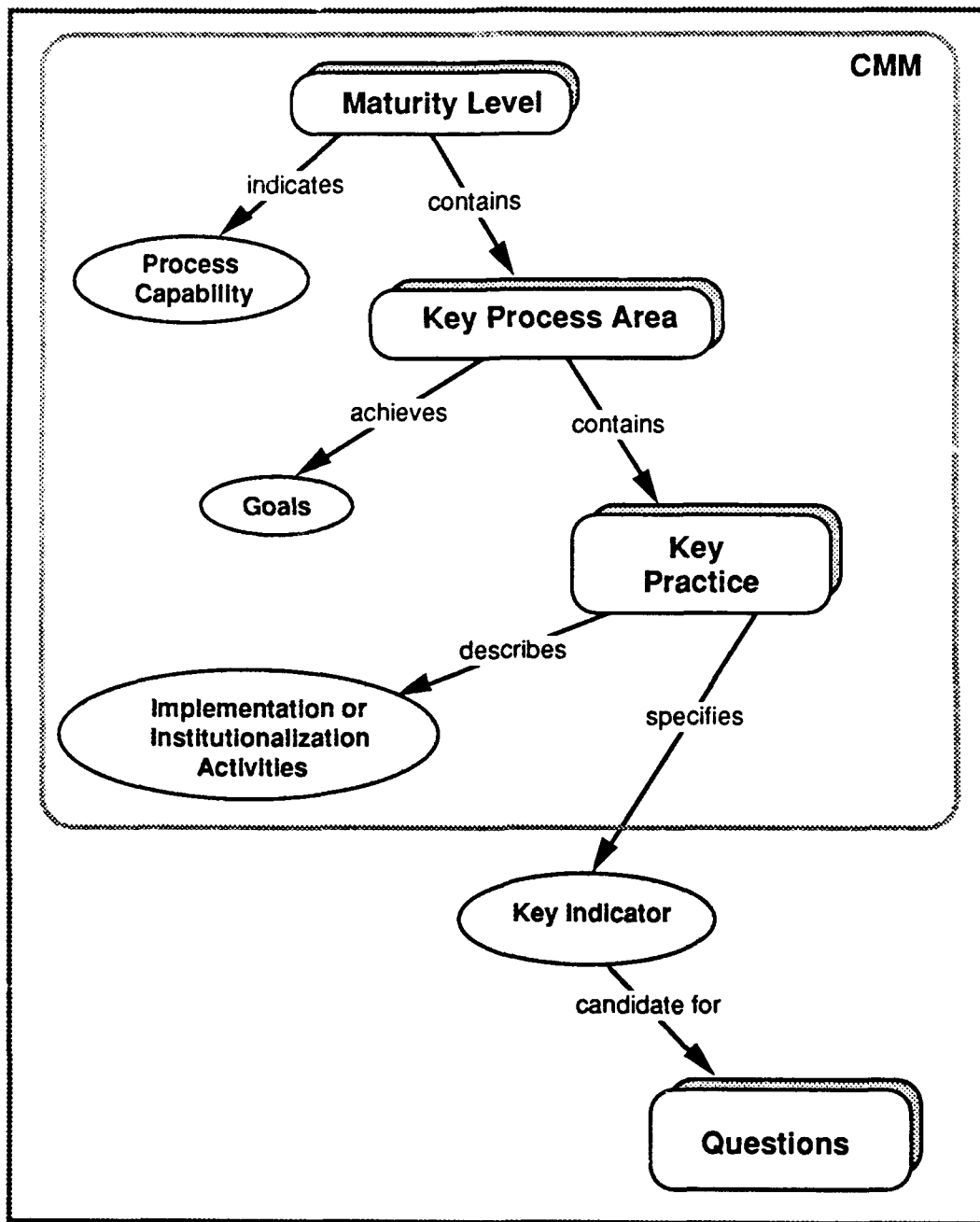


Figure 2.1 The Structure of the Capability Maturity Model

Introducing the Capability Maturity Model

- Key process areas* Each maturity level is composed of key process areas. Each key process area is a cluster of related activities that when performed collectively achieve a set of goals important for establishing process capability at that maturity level. For example, one of the key process areas for Level 2 is Software Project Planning.
- Goals* The goals summarize the intent of the key practices in the key process area. An example of a goal from the Software Project Planning key process area is "A plan is developed that appropriately and realistically covers the software activities and commitments."
- Key practices* Each key process area is described in terms of key practices that, when implemented, help to satisfy the goals of that key process area. For example, one of the key practices from the Software Project Planning key process area is "Senior management reviews and approves all commitments made to individuals and groups external to the organization."

2.4 What Do the Maturity Levels Mean?

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

Each maturity level provides a layer in the foundation for continuous process improvement. Each maturity level comprises a set of process goals that, when satisfied, stabilize an important component of the software process. Achieving each level of the maturity model establishes a different component in the software process, resulting in an increase in the process capability of the organization.

Introducing the Capability Maturity Model

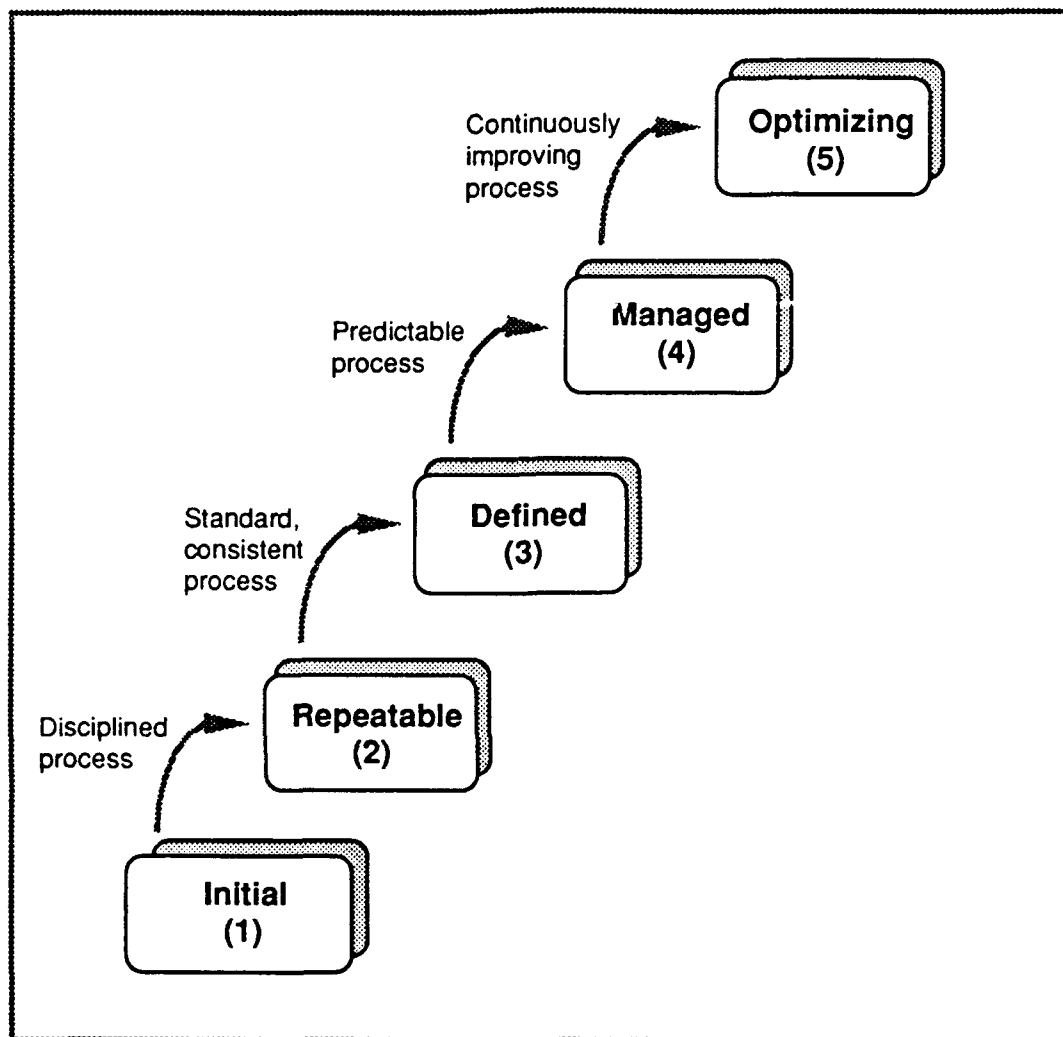


Figure 2.2 The Five Levels of Software Process Maturity

2.4.1 Level 1 - The Initial Level

At the Initial level, the organization typically does not provide a stable environment for developing and maintaining software. When sound management practices are lacking within an organization, the benefits of

Introducing the Capability Maturity Model

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 (i.e., the process is ad hoc). Schedules, budgets, functionality, and product quality are generally unpredictable. Performance depends on the individual capabilities of the staff and managers and varies 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.

2.4.2 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

Introducing the Capability Maturity Model

observed on previous projects. The software managers for a project 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.

2.4.3 Level 3 - The Defined Level

At the Defined level, the standard process for developing and maintaining software across the organization is documented, including both software engineering and software management processes, and they are integrated into a coherent whole. Processes established at Level 3 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 its software processes. There is a permanent organizational focus on the software process; a software engineering process group facilitates process definition and improvement efforts (Fowler90). An organization-wide training program is implemented to ensure that the staff 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

Introducing the Capability Maturity Model

project uses a peer review process to enhance product quality. Because the software process is well defined, management has good insight into technical progress on all projects.

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.

2.4.4 Level 4 - The Managed Level

At the Managed level, the organization sets quantitative quality goals for software products and processes. 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 metrics at Level 4. These metrics establish the quantitative foundation for evaluating the projects' processes and products.

Projects achieve control over their products and processes by narrowing the variation in their process performance to within acceptable quantitative boundaries. Meaningful variations in process performance can be distinguished from random variation (noise), particularly within established product lines. To reduce the process variation caused by 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.

Introducing the Capability Maturity Model

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.

2.4.5 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 of new technologies. Innovations that exploit the best software engineering practices are identified.

Project teams in Level 5 organizations analyze process performance to determine the causes of defects. They evaluate their software processes to prevent known types of defects from recurring, and they disseminate lessons learned to other projects.

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.

2.5 What Are Key Process Areas?

Figure 2.3 lists the key process areas for each maturity level in the CMM.

The key process areas are building blocks that indicate the areas an organization should focus on to improve its software process. Key process areas identify the issues that must be addressed in order to achieve a maturity level.

By definition, key process areas reside at a single maturity level.

Introducing the Capability Maturity Model

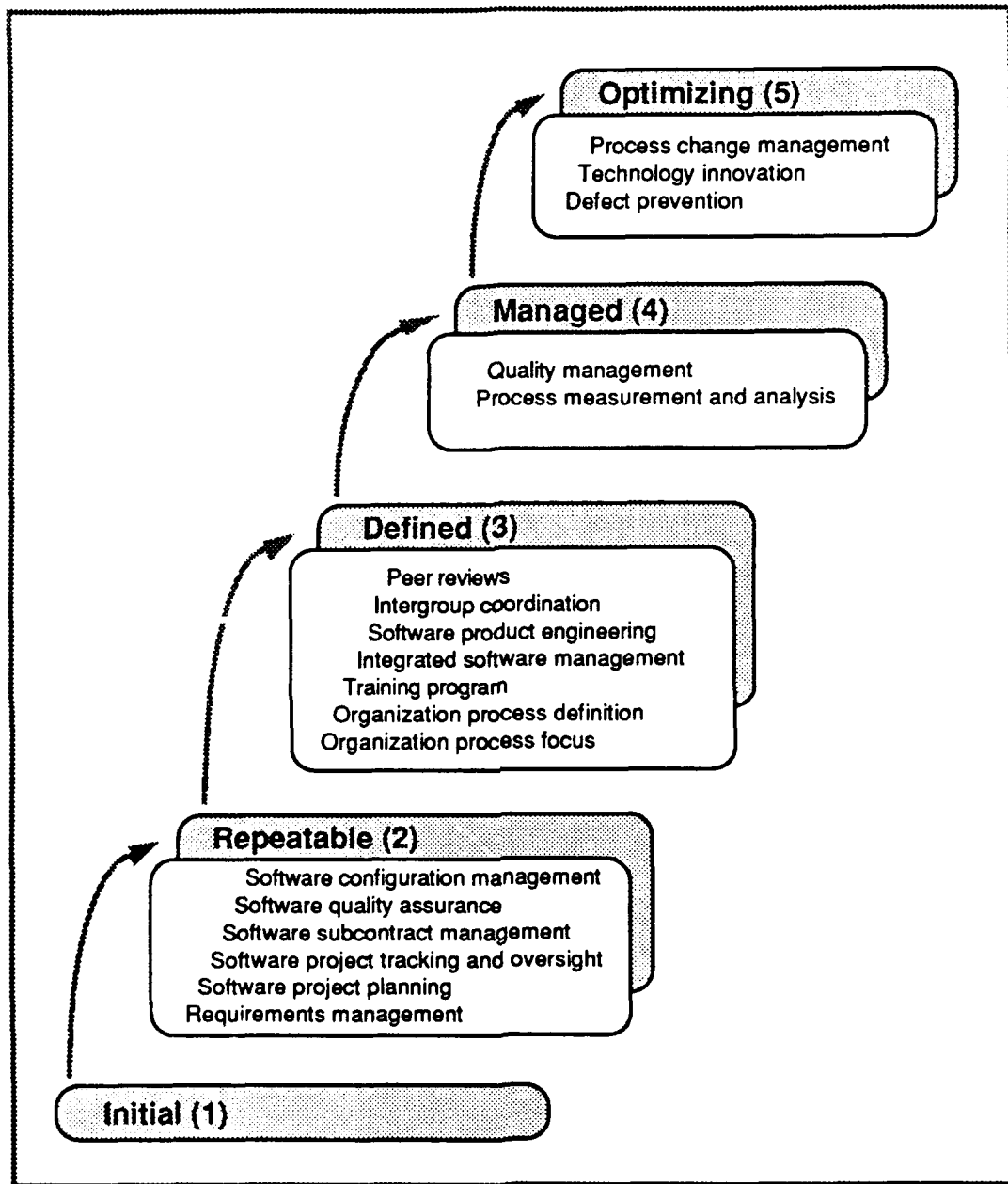


Figure 2.3 The Key Process Areas by Maturity Level

2.6 What Are Goals?

The goals can be used to determine whether an organization or project has effectively implemented a key process area. The goals signify the scope, boundaries, and intent of each key process area. In adapting the key practices of a key process area to a specific project situation, the goals can be used to determine whether the adaptation is a faithful rendering of the practices. Similarly, when assessing or evaluating alternative ways to implement a key process area, the goals can be used to determine if the alternatives satisfy the intent of the key process area.

2.7 What Are the Key Practices?

The key practices are the specific, lowest-level details of the CMM. The key practices are the policies, procedures, and activities that most significantly contribute to the institutionalization and implementation of the key process area. They are a working definition of the key process area.

The key practices are also the link between the CMM and the maturity questionnaire. Specific questions in the maturity questionnaire relate to specific key practices.

Although many practices contribute to success in developing effective software, the key practices were identified because of their effectiveness in improving an organization's capability in a particular key process area.

Introducing the Capability Maturity Model

Since a primary use of these key practices is to provide guidance to a group such as the software engineering process group, the details of the key practices are important. When used in software process assessments and software capability evaluations, it is expected that the assessment and evaluation teams will focus on the goals, the top-level key practices, and other key practices specifically identified by empirical studies and industry experiences as providing the best insight into whether the goals have been satisfied.

2.8 What Are the Common Features?

The key practices in each key process area are organized by a set of common features. The common features are the attributes that you would want to investigate to determine if an implementation of a key process area is effective, repeatable, and lasting. The common features also group and order the key practices in a sequence helpful for organizations using them.

The key practices for each key process area are grouped into five common features:

- ☐ Commitment to Perform,
- ☐ Ability to Perform,
- ☐ Activities Performed,
- ☐ Monitoring Implementation, and
- ☐ Verifying Implementation.

The Activities Performed by projects provide the largest category of key practices because they describe the actual implementation of the key process area. Key practices under the other common features address what must be done to support and institutionalize the key process area.

Introducing the Capability Maturity Model

<i>Commitment to Perform</i>	Commitment to Perform specifies the actions the organization must take to ensure that the process is established and will endure. Commitment to Perform typically involves senior management sponsorship and establishing policies.
<i>Ability to Perform</i>	Ability to Perform specifies the preconditions that must exist in the project or organization in order to implement the process competently. Ability to Perform typically involves training, special skills, and special tools.
<i>Activities Performed</i>	Activities Performed specifies the steps that must be performed to effectively establish the key process area. Activities Performed typically involve establishing plans and procedures, performing the work, and verifying and correcting the results of the work.
<i>Monitoring Implementation</i>	Monitoring Implementation specifies the steps that must be performed to measure the process, analyze the measurements, and take action based on the results.
<i>Verifying Implementation</i>	Verifying Implementation specifies the steps that must be performed to guide and ensure that the activities are performed in compliance with the process that has been specified. Verification typically encompasses reviews and audits.

Introducing the Capability Maturity Model

3 Interpreting the Key Practices

3.1 How Should the Key Practices Be Used?

Our intention in setting down the key practices is not to require or espouse a specific model of the software life cycle, a specific organizational structure, a specific separation of responsibilities, or a specific approach to development. Our intention, rather, is to ensure that the essential elements of an effective software process are satisfied.

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

3.2 What Conventions Are Used in the Key Practices?

Although the key practices are meant to be independent of any particular implementation, we must use specific terms and examples to make our statements of the key practices understandable. Because we want to be internally consistent in our use of terms, all of the key practices use a particular set of conventions for roles, responsibilities, relationships, products, and activities.

Interpreting the Key Practice Statements

On the following pages, the conventions that are used in stating the key practices are described. You should be aware of them so that you can map the conventions to your own organization, project, and situation.

The glossary in Appendix B contains definitions of terms beyond those discussed on the following pages.

3.3 The Goals of Each Key Process Area Are the Essential Criteria

All of the key practices in a key process area must be interpreted in light of a project's or organization's specific situation.

The key practices define a comprehensive process that is appropriate for large, complex systems for critical applications. Instantiating the key practices for other systems may mean tailoring them. For example, the key practices require a software development plan. The level of detail and the formality of that plan may be very different for projects of different sizes, levels of complexity, and degrees of criticality.

The pertinent question is whether the implementation of the key practices satisfies the goals of the key process area. This requires professional judgment. Therefore, projects and organizations must interpret the key practices in a thoughtful and justifiable manner. They should document the rationale for tailoring the key practices for a given situation.

3.4 The CMM Does Not Imply a Particular Life-Cycle Model

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

The term "stage" is used to refer to a partition of the software effort, but we do not mean to tie the term to any specific life-cycle model. As it is used in the key practices, "stage" can mean rigidly sequential stages or overlapping and iterative stages. For those who use the waterfall life-cycle model, the term "stage" is equivalent to "phase."

3.5 The CMM Does Not Imply a Specific Software Technology

In the same vein, the key practices neither require nor preclude specific software technologies, such as prototyping or reusing software requirements, design, code, or other elements.

3.6 The CMM Does Not Require a Specific Set of Documents

The key practices do call for a number of process-related documents, each covering specific areas of content. However, the key practices do not require

Interpreting the Key Practice Statements

a one-to-one relationship between the documents named in the practices and the products of an organization or project. Nor is there an intended one-to-one relationship to documents specified in DoD standards, such as DoD-STD-2167A. The key practices require only that the contents of these documents be part of the organization's or project's written products.

The contents of a document referred to in the key practices could be just one part of a larger document. For example, an organization might have a process development plan that includes the essentials of the process improvement plan.

Alternatively, the contents of a document referred to in the key practices could be distributed over a set of documents that differ from the set named in the key practices. For example, a project might develop three documents, a software development plan, a software management plan, and a project work breakdown structure, to satisfy the key practices for a software project's statement of work and a software development plan.

3.7 The CMM Does Not Espouse Particular Roles or Organization Structure

Figure 3.1, on the next page, shows the structure of the software organization that we use as our example. This example is used to illustrate the roles, responsibilities, and relationships inherent in the key practices. This organization structure is neither required nor is it necessarily the best structure for implementing a software process for your organization.

Interpreting the Key Practice Statements

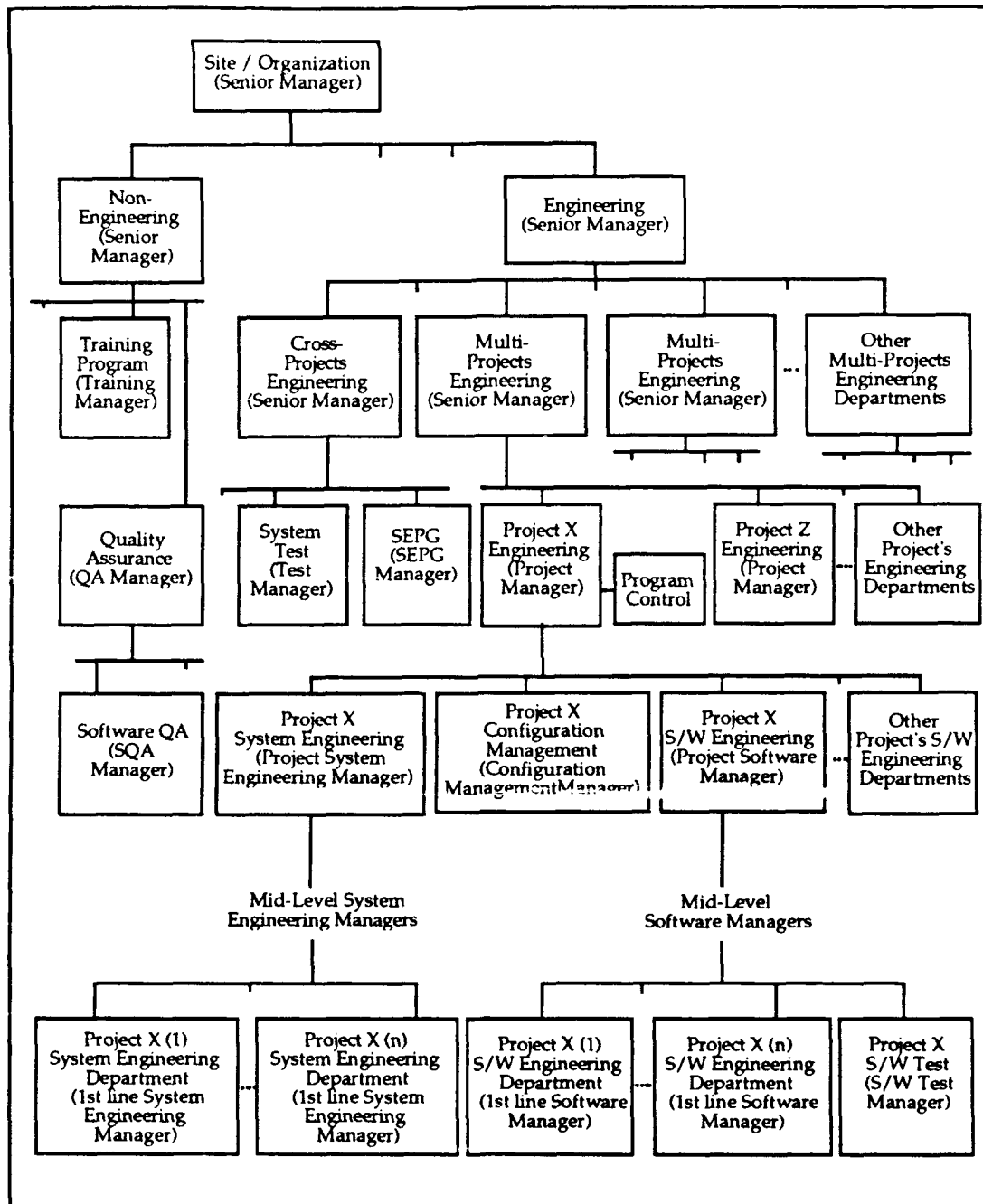


Figure 3.1 Conceptual Software Organization Used in the CMM

3.8 The Example Organization Includes These Roles

In the key practices, we use the following roles. The first five are shown on the example of an organization chart in Figure 3.1.

Senior manager A senior manager is at a high enough level that his or her primary focus is expected to be the long-term technical and business vitality of the organization and company, rather than short-term concerns and pressures of particular contracts or projects. A senior manager has responsibility for more than one project. The resources for long-term improvement of the software process (e.g., a software engineering process group) would generally be provided by, and protected by, a senior manager.

Project manager A project manager is the person who has total business responsibility for an entire project. The project manager is the person who is ultimately responsible to the customer.

In a project-oriented organizational structure, most of the people working on a project would report to the project manager, although some disciplines might have a matrixed reporting relationship. In a matrixed organizational structure, it may be only the business staff who reports to the project manager. The engineering groups would then have a matrixed reporting relationship.

Interpreting the Key Practice Statements

Project software manager

A project software manager is the person who has total responsibility for all the software activities for the entire project. This is the person whom the project manager deals with on issues such as software commitments. The software development people on a project would report to the project software manager, although some activities such as tools development might have a matrixed reporting relationship.

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

Mid-level software manager

A mid-level software manager reports directly, or indirectly through another manager, to the project software manager. The mid-level software manager has direct management responsibility for other software managers, including responsibility for career planning, personnel, and salary.

First-line software manager

A first-line software manager has direct management responsibility for the staff and activities of a single department of software engineers and other related staff, including responsibility for career planning, personnel, and salary.

Interpreting the Key Practice Statements

Software task leader

A software task leader leads a technical team for a specific software task. The software task leader is responsible for providing technical direction to the staff who are working on the task, including him- or herself. The software task leader usually reports to the same first-line software manager as the other people who are working on the task.

Software engineering staff

The software engineering staff are the technical people, such as analysts, programmers, and engineers, who are not managers and who perform the software development and maintenance activities. The software task leaders are also members of the software engineering staff. For example, members of the software engineering staff analyze and specify software requirements and design, code, and test the software.

A similar breakout of roles exists for other engineering groups such as system engineering.

3.9 The Example Organization Includes These Groups

Software engineering group

The software engineering group is the collection of departments, managers, and staff who are responsible for the software development activities of a project. The software quality assurance group, the software configuration management group, and the software engineering process group are not part of the software engineering group.

Interpreting the Key Practice Statements

*Software
engineering
process group*

The software engineering process group includes the specialists who coordinate the activities for defining and improving the organization's software process. The software engineering process group may be either a clearly distinguishable organizational entity or a group that is distributed among various other software groups and committees.

*System
engineering
group*

The system engineering group is the collection of departments, managers, and staff who have responsibility for specifying the system requirements; allocating the system requirements to the hardware, software, firmware, and human components; specifying the interfaces between the hardware, software, firmware, and human components; and monitoring the design and development of these components to ensure conformance with their specifications.

System test group

The system test group is the collection of departments, managers, and staff who have responsibility for planning and performing the independent system testing of the software to determine whether the software product satisfies its allocated requirements.

*Software quality
assurance group*

The software quality assurance group is the collection of departments, managers, and staff who plan and implement the quality assurance activities to ensure the process steps and standards are followed.

Interpreting the Key Practice Statements

*Software
configuration
management
group*

The software configuration management group is the collection of departments, managers, and staff who have responsibility for planning, coordinating, and implementing the formal configuration management activities for the software project.

Training group

The training group is the collection of departments, managers, and staff who have responsibility for planning and coordinating the management and technical training for the organization. This group usually prepares and conducts most of the training courses.

3.10 The Actual Assignment of Roles May Vary

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

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

On a slightly larger project, one person might be the system engineering first-line manager, the project system engineering manager, and the project manager while another person might be both the first-line software manager and the project software manager. These two managers might be

in the same second-line organization or in different second-line organizations.

On a large project, each of these roles would likely be filled by different people.

3.11 The Composition of Groups May Vary

Similarly, a "group" might be a single person within a department or it may include people from many departments, again depending on the size of the project and other considerations. For example, on a small project, the software configuration management group might consist of a single software engineer dedicated half-time and a first-line software manager dedicated part-time.

3.12 Independence of Responsibilities May Be a Concern on Small Projects

On a small project, when one person performs several roles, care must be taken to ensure that the key practices that call for independence are followed. The key practices call for independence when technical or organizational biases may affect the quality or risks associated with the project. For example, two key practices dealing with independence are:

- ☐ The system test cases and test procedures are prepared by a group that is separate from the group that developed the software.
- ☐ The software quality assurance group has a reporting channel to senior management that is independent of the software engineering group, software-related groups, and the project manager.

Interpreting the Key Practice Statements

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

The independence of the software quality assurance group is necessary so its members can perform their jobs without being influenced by schedule and cost pressures. Ensuring effective operational independence without the organizational independence is difficult. For example, an employee reporting to the project manager may be reluctant to stop a delivery even though serious noncompliance issues exist.

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

3.13 The CMM Allows Flexibility

To provide a complete set of valid principles that apply to a wide range of situations, we have intentionally stated some of the key practices to allow for flexibility. Throughout the key practices, you will find nonspecific phrases like "regular reviews," "affected groups," "as appropriate," and "updated as necessary." These phrases may have different meanings for two projects in a single organization or even for one project at different points in its life cycle. Each project or organization must clarify these phrases for its specific situation.

Clarifying these phrases requires you to consider the overall context in which they are used. The pertinent question is whether the specific interpretation of one of these phrases meets the goals of the key process area. Professional judgment must be used to determine whether the goals

have been achieved. The glossary in Appendix B may provide guidance in interpreting these and other phrases in the key practices.

3.14 The Organization's and Projects' Processes Must Have a Common Basis

A standard software process establishes a consistent way of performing the software activities across the organization and is essential for the organization's long-term stability and improvement. Flexibility to adjust the process for specific needs of the projects is essential for satisfying the projects' needs.

The key practices are presented using terms that relate to a conceptual structure of process components and activities that support these two assertions. This conceptual structure is depicted in Figure 3-2 and its key elements follow.

Interpreting the Key Practice Statements

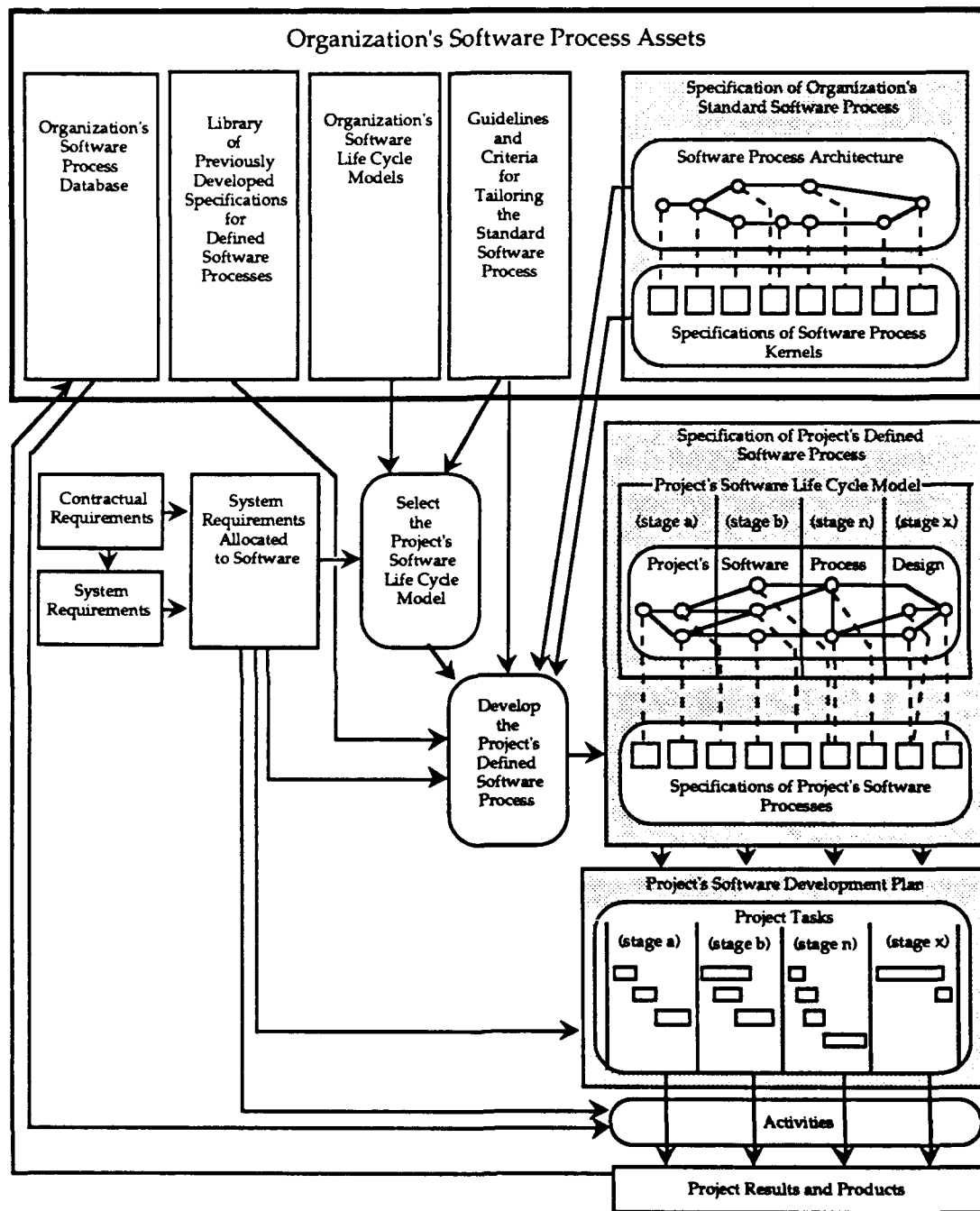


Figure 3.2 Conceptual Software Process Structure Used in the CMM

Interpreting the Key Practice Statements

<i>Organization's software process assets</i>	The organization establishes a set of software process assets as shown in Figure 3.2. These software process assets consist of process artifacts maintained by an organization for use by the projects in defining, maintaining, and implementing their defined software process. An organization may have more than one set of software process assets if, for example, it develops two significantly different types of applications.
<i>Organization's standard software process</i>	The organization's standard software process includes a framework of a start-to-end software process and the specifications of software process kernels for the processes that are common across the organization's projects. It forms the basis for the projects' defined software processes. The standard software process provides continuity in the organization's process activities and is the reference for the measurements and long-term improvement of the organization's software process.
<i>Software process architecture</i>	The software process architecture is a high-level (i.e., summary) specification of the organization's standard software process. It describes and orders the software tasks and activities that are common across the projects in the organization.
<i>Software process kernels</i>	The software process kernels, which populate the software process architecture, specify the fundamental tasks and activities used in building a project's software process. These software process kernels are used by projects as the basis for developing their software process specifications.

Interpreting the Key Practice Statements

<i>Project's defined software process</i>	The defined software process for a project is a well-characterized and understood software process, defined in terms of software standards, procedures, tools, and methods. The project's defined software process is developed to accommodate the system requirements allocated to software by selecting a software life-cycle model and tailoring the organization's standard software process according to organizational guidelines and criteria. It provides the basis for controlling, coordinating, and improving the activities of staff and managers performing the project's tasks. It is possible to have more than one defined software process (e.g., for the operational software and for the test support software) or to have one defined software process for two or more similar projects.
<i>Software life-cycle model</i>	The project's software life-cycle model is a top-level plan for the software effort. It specifies the primary stages of the software life cycle and the relationships between these stages.
<i>Stages</i>	A stage is a partition of the software effort. It is a manageable size and represents a meaningful and measurable set of related tasks that are performed by the project.
<i>Software process design</i>	The software process design is a high-level (i.e., summary) specification of a project's software process for the selected software life cycle. It describes the software tasks and activities that are performed, and it identifies the major software process components and their connections to each other.

Interpreting the Key Practice Statements

Project's software process specifications The specifications of the project's software process are the operational definitions of the software process components identified in the project's defined software process.

Tasks The work to be performed is broken down into tasks, which provide visible checkpoints for management to determine the status of the project.

Activities Activities includes all the things the staff and managers do to perform the tasks of the project and organization.

At the organizational level, the process needs to be specified, controlled, and changed in a formal manner. At the project level, the emphasis is on the usability of the process specification and the value it adds to the project - the formality and control of the project's process specifications depend on the project characteristics.

Interpreting the Key Practice Statements

4 Using the Key Practice Pages

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

Each key process area contains:

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

The key practices themselves are grouped into the five common features (commitment to perform, ability to perform, activities performed, monitoring implementation, and verifying implementation) and are presented in a hierarchical format, as shown in Figure 4.1, an example page from the key practices. The key practices include:

Top-level key practices

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

Using the Key Practice Pages

Subordinate key practices Subordinate key practices are listed beneath the top-level key practices and describe what we would expect to find implemented for the top-level practices. The subordinate key practices can be used to help determine whether or not the top-level key practices are implemented satisfactorily.

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

When the subordinate key practices or the supplementary information underneath a top-level key practice extends to another page, the number of the top-level key practice is shown in parentheses at the start of the new page to indicate that the information on that page is a continuation of the top-level key practice on the previous page.

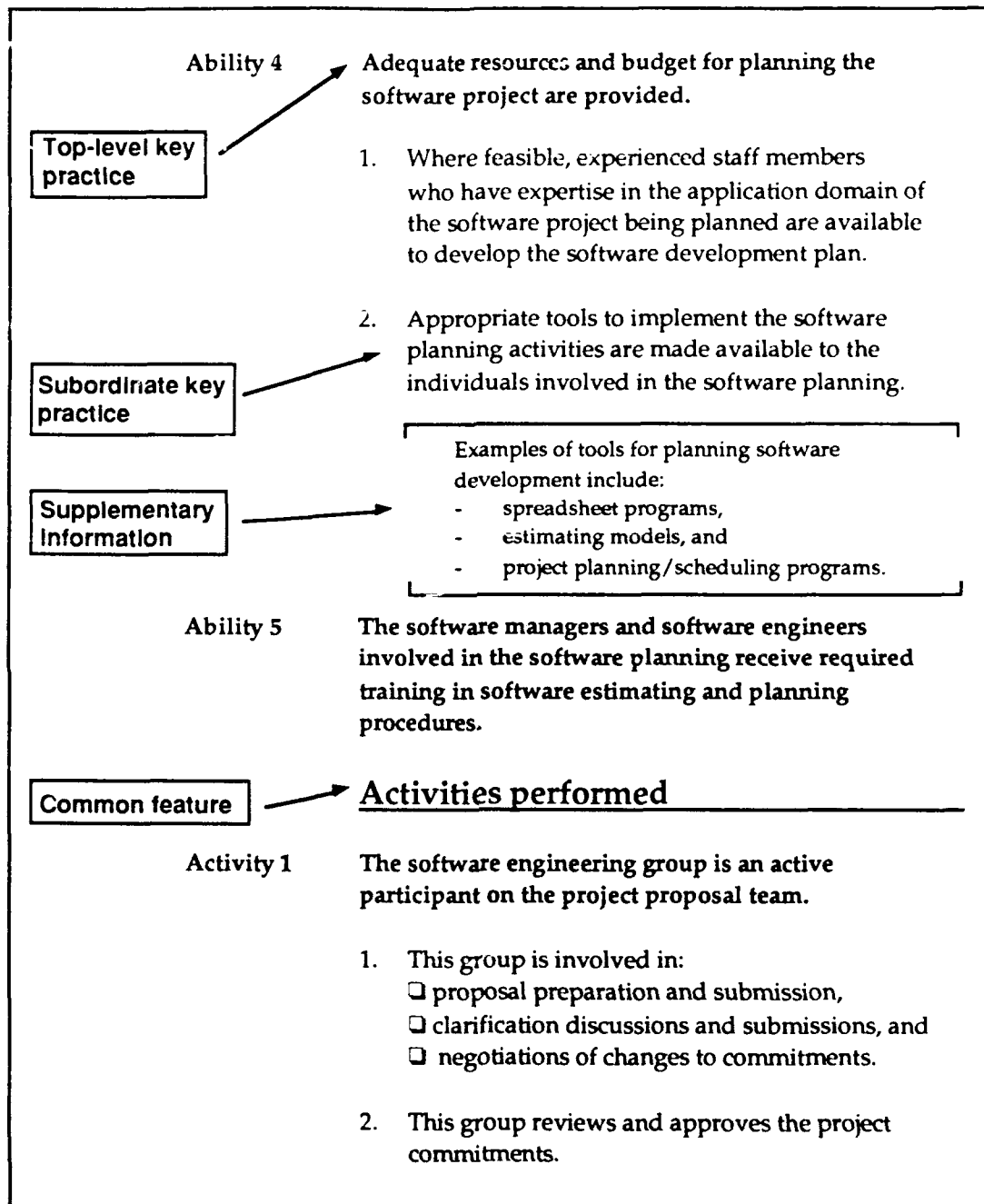


Figure 4.1 Example of Key Practice Statements

Using the Key Practice Pages

Requirements Management

a key process area for Level 2: Repeatable

Requirements management involves establishing and maintaining an understanding and agreement with the customer on the requirements for the software throughout the software life cycle. The agreements cover both the technical requirements for the software and the nontechnical requirements, such as delivery dates for the software. The agreements form the basis for estimating, planning, performing, and tracking the project's software activities.

Goals

Goal 1

The system requirements allocated to software provide a clearly stated, verifiable, and testable foundation for software engineering and software management.

The system requirements allocated to software are referred to as "allocated requirements" in these practices.

The allocated requirements are the subset of the system requirements that are to be implemented in the software components of the system. The elaboration and refinement of the allocated requirements result in the software requirements specification and are a primary input to the software development plan.

Goal 2

The allocated requirements define the scope of the software effort.

- Goal 3** **The allocated requirements and changes to the allocated requirements are incorporated into the software plans, products, and activities in an orderly manner.**

Commitment to perform

- Commitment 1** **The organization follows a written policy for managing the project requirements that determine and bound the software activities.**

This policy requires that:

1. The allocated requirements are documented.
2. The allocated requirements are reviewed and agreed to by:
 - ☐ the software managers,
 - ☐ the software task leaders, and
 - ☐ other affected groups.

Examples of affected groups include:

- system test,
 - software quality assurance,
 - software configuration management, and
 - documentation support.

3. The software plans, products, and activities are changed when the allocated requirements change.

Ability to perform

Ability 1

For each project, responsibility is established for analyzing the system requirements and allocating them to hardware, firmware, software, and manual processes.

This responsibility covers:

1. Managing and documenting the system requirements and their allocation throughout the project's life cycle.
2. Controlling changes to the system requirements and their allocations.

An example of assigning this responsibility is to establish a system design team with members from the appropriate engineering groups.

Ability 2

Adequate resources and funding are provided for managing the allocated requirements.

1. Staff members who have experience and expertise in the application area and in software engineering support management of the allocated requirements.
2. Appropriate tools to support the activities for managing requirements are made available.

(Ability 2)

Examples of tools for managing requirements include:

- spreadsheet programs,
- tools for configuration management,
- tools for traceability, and
- tools for test management.

Activities performed

Activity 1

The allocated requirements are documented in a consistent format and are clearly stated, verifiable, and testable.

The allocated requirements include:

1. The agreements, conditions, and contractual terms that affect and determine the software effort.

Examples of agreements, conditions, and contractual terms include:

- deliverables,
- delivery dates,
- milestones,
- programming languages, and
- software engineering environments.

2. The technical requirements for the software.
3. The interface requirements, including the requirements for the external software interfaces to:
 - ☐ hardware,
 - ☐ other software systems, and
 - ☐ human interfaces.

(Activity 1)

4. The criteria that will be used to evaluate the software products for acceptance.

Activity 2

The software engineering group reviews and agrees to the allocated requirements before they are incorporated into the software efforts.

1. Incomplete and missing allocated requirements are identified.
2. The allocated requirements are analyzed to determine whether they are:
 - ☐ feasible and appropriate to implement in software,
 - ☐ clearly stated,
 - ☐ consistent with each other, and
 - ☐ verifiable.
3. The allocated requirements that are problems are identified and reviewed with the team responsible for analyzing and allocating system requirements, the customer, and the end users, as appropriate, and appropriate changes are made.

The end users referred to in these practices are the customer-designated end users or representatives of the end users.

4. The allocated requirements are reviewed and agreed to by:
 - ☐ the software task leaders,
 - ☐ the first-line software managers,
 - ☐ the mid-level software managers, and
 - ☐ the project software manager.
5. Commitments and changes to commitments resulting from the allocated requirements are negotiated with the affected groups and individuals.

(Activity 2)

Refer to the software project planning and software project tracking and oversight key process areas for practices covering making and changing commitments.

Activity 3

The allocated requirements form the basis for the software plans, products, and activities.

The allocated requirements:

1. Are maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

2. Are the basis for developing the software requirements specification and the software development plan.

Activity 4

Changes to the allocated requirements are appropriately reviewed and incorporated into the software efforts.

1. The software engineering group and other software-related groups participate or are appropriately represented in evaluating and approving changes to the allocated requirements.
2. The changes to the allocated requirements are documented and maintained under configuration management.

(Activity 4)

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

3. Changes to the allocated requirements are reviewed and agreed to by the software task leaders, first-line software managers, mid-level software managers, and project software manager before they are incorporated into the software plans, products, and activities.
4. The impact to existing commitments is assessed and changes are negotiated as appropriate.
 - ☐ Changes to contractual commitments are reviewed and approved by senior management.
 - ☐ Changes to commitments within the organization are arrived at according to a documented commitment review procedure.

Refer to the software project planning and software project tracking and oversight key process areas for practices covering making and changing commitments.

5. Changes to the software plans, products, and activities resulting from changes to the allocated requirements are:
 - ☐ identified,
 - ☐ evaluated,
 - ☐ communicated to the groups and individuals who may be affected, and
 - ☐ tracked.

Monitoring implementation

Monitor 1

Measurements are made and used to quantify and evaluate the cost and schedule status of the activities for managing requirements.

1. The status of each of the allocated requirements is tracked; status and deviations that require management action are reported to the appropriate managers.
2. Change activity for the allocated requirements is tracked; status and issues that require management action are reported to the appropriate managers. The following are tracked:
 - ☐ total number of changes, over time, from each major source (e.g., customer, end users, software engineering group, system engineering group, and system test group), and
 - ☐ number of changes, over time, including total number of changes proposed, open, approved, and incorporated into the system baseline.

Verifying implementation

Verification 1

The activities for managing requirements are reviewed with senior management on a regular basis.

Refer to the software project tracking and oversight key process area for practices covering the oversight reviews with senior management.

Verification 2

The activities for managing requirements are reviewed with the project manager on a regular basis.

(Verification 2)

Refer to the software project tracking and oversight key process area for practices covering the status/coordination reviews with the project manager.

Verification 3

The software quality assurance group reviews and audits the project's activities and products for managing requirements, and reports results as appropriate.

At a minimum, the reviews and audits confirm that:

1. The allocated requirements are appropriately reviewed and analyzed and issues are appropriately resolved before the software engineering group commits to them.
2. The software plans, products, and activities are appropriately revised when the allocated requirements change.
3. Changes to commitments resulting from changes to the allocated requirements are made according to a documented commitment review procedure.

Refer to the software quality assurance key process area for practices covering the quality assurance reviews and audits.

Software Project Planning

a key process area for Level 2: Repeatable

Software project planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work. A plan is established to address the commitments to the customer according to the resources, constraints, and capabilities of the project. The plan provides the basis for initiating the software effort and managing the progress of the work.

Goals

Goal 1	A plan is developed that appropriately and realistically covers the software activities and commitments.
Goal 2	All affected groups and individuals understand the software estimates and plans and commit to support them.
Goal 3	The software estimates and plans are documented for use in tracking the software activities and commitments.

Commitment to perform

Commitment 1	A project software manager is designated to be responsible for negotiating commitments and developing the project's software development plan.
Commitment 2	The organization follows a written policy for planning a software project.

(Commitment 2) This policy requires that:

1. The software engineering group's commitments are negotiated between:
 - ☐ the project manager,
 - ☐ the project software manager,
 - ☐ the mid-level software managers, and
 - ☐ the first-level software managers.
2. Involvement of other project groups in the software activities is negotiated with these groups and is documented.
3. All affected groups review and agree to the project's:
 - ☐ software size estimates,
 - ☐ cost estimates,
 - ☐ schedules, and
 - ☐ other commitments.

Examples of affected groups include:

- software engineering (including all subgroups, such as software requirements analysis, as well as the software task leaders),
- software estimating,
- system engineering,
- system test,
- software quality assurance,
- software configuration management,
- finance, and
- documentation support.

4. Senior management and the customer review and approve all customer commitments.

- (Commitment 2)** 5. The project's software development plan is documented and controlled.

Ability to perform

Ability 1

A statement of work for the software effort is documented and approved.

1. The statement of work covers:
 - ☐ technical goals and objectives,
 - ☐ identification of users and customers,
 - ☐ imposed standards,
 - ☐ assigned responsibilities,
 - ☐ cost and schedule constraints and goals,
 - ☐ resource constraints and goals, and
 - ☐ other development constraints and goals.
2. The statement of work is reviewed and approved by:
 - ☐ the first-line software managers,
 - ☐ the mid-level software managers,
 - ☐ the project software manager,
 - ☐ the project manager, and
 - ☐ the system engineering group.
3. The statement of work is maintained under configuration management.

(Ability 1)

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Ability 2

The system requirements allocated to software are documented and approved.

The system requirements allocated to software are referred to as "allocated requirements" in these practices.

The allocated requirements are the subset of the system requirements that are to be implemented in the software components of the system. The elaboration and refinement of the allocated requirements result in the software requirements specification and are a primary input to the software development plan.

Refer to the requirements management key process area for practices covering the allocated requirements.

Ability 3

Responsibilities for developing the software development plan are assigned.

1. The project software manager, directly or by delegation, coordinates the project's software planning.
2. Responsibilities for the software products and activities are partitioned and assigned to the mid-level software managers and first-line software managers in a traceable, accountable manner.

Ability 4 **Adequate resources and budget for planning the software project are provided.**

1. Where feasible, experienced staff members who have expertise in the application domain of the software project being planned are available to develop the software development plan.
2. Appropriate tools to implement the software planning activities are made available to the individuals involved in the software planning.

Examples of tools for planning software development include:

- spreadsheet programs,
- estimating models, and
- project planning/scheduling programs.

Ability 5 **The software managers and software engineers involved in the software planning receive required training in software estimating and planning procedures.**

Activities performed

Activity 1 **The software engineering group is an active participant on the project proposal team.**

1. This group is involved in:
 - ☐ proposal preparation and submission,
 - ☐ clarification discussions and submissions, and
 - ☐ negotiations of changes to commitments.
2. This group reviews and approves the project commitments.

(Activity 1)

Examples of project commitments include:

- the project technical goals and objectives;
- the system and software technical solution;
- the software budget, schedule, and resources; and
- the software standards and procedures.

Activity 2 **The software planning is initiated in the early stages of, and in parallel with, the overall project planning.**

Activity 3 **The software engineering group actively participates in the overall project planning throughout the project's life.**

The software engineering group reviews and approves the project-level plans.

Activity 4 **Senior management reviews and approves all commitments made to individuals and groups external to the organization.**

Activity 5 **A software life-cycle model with predefined stages of manageable size is identified or defined.**

The software life-cycle model is selected based on the technical characteristics of the software and the cost and schedule goals.

Examples of software life-cycle models include:

- waterfall,
- overlapping waterfall,
- spiral,
- serial build, and
- single prototype/overlapping waterfall.

Activity 6

The project's software development plan is developed according to a documented procedure.

This procedure requires that:

1. The software development plan is based on and conforms to:
 - ☐ the customer and project standards,
 - ☐ an approved statement of work, and
 - ☐ the approved specification of allocated requirements.
2. Plans for groups involved in the software activities are negotiated with those groups, the support efforts are budgeted, and these agreements are documented.
3. Plans for involvement of the software engineering group in the activities of other project groups are negotiated with those other groups, the support efforts are budgeted, and the agreements are documented.
4. The groups who are potentially involved in or affected by the activities and products covered in the software development plan review and agree with the plan and commit their support, as appropriate.

Examples of affected groups include:

- software engineering (including all subgroups, such as software design, as well as the software task leaders),
- software estimating,
- system engineering,
- system test,
- software quality assurance,
- software configuration management,
- finance, and
- documentation support.

- (Activity 6)** 5. The software development plan is maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

- Activity 7** **The software development plan covers (directly or by reference) the plan for the software activities.**

Specifically, this plan covers:

1. Project purpose, scope, goals, and objectives.
2. Identification and/or description of the project's selected software processes.
3. Identification of the selected procedures, methods, and standards for software management and development.
4. Identification of software products to be developed, including:
 - ☐ major products for internal use of the software engineering group,
 - ☐ products for use by other project groups, and
 - ☐ products for delivery to the external customer.
5. Size estimates of the software products.
6. Staff resource estimates.
7. Software project schedules, including identification of milestones.

(Activity 7) 8. Identification and assessment of the project's software risks.

Activity 8 Software products and software process specifications that are needed to establish and maintain stability of the software activities are explicitly identified to be controlled project baseline items.

Refer to the software configuration management key process area for practices covering the baseline items.

Activity 9 Estimates for the size of the software products are derived according to a documented procedure.

This procedure requires that:

1. Size estimates are made for all major software products and activities.

Examples of types of products and activities for which size estimates are made include:

- operational software and support software,
- deliverable and nondeliverable products,
- software and nonsoftware products (e.g., documents), and
- product development and product verification activities (e.g., executing test cases).

2. Software products are decomposed into constituent elements to the granularity needed for the estimating objectives.
3. Historical data are used where available.
4. Size estimating assumptions are documented.

(Activity 9)

5. Size estimates are documented, reviewed, and agreed to.

Examples of groups and individuals who review and agree to size estimates include:

- the software task leaders,
- the first-line software managers,
- the mid-level software managers,
- the project software manager,
- the project manager,
- the system test group, and
- the system engineering group.

Activity 10

Estimates for software development resources and costs are derived according to a documented procedure.

This procedure requires that:

1. Estimates for software development resources and costs are related to the size estimates of the software products.
2. Objective productivity data (historical and/or current) are used for the estimates, when available; sources and rationale for these data are documented.
 - ☐ The productivity and cost data are from the organization's projects when possible.
 - ☐ The productivity and cost data take into account the activities and other applicable costs that go into making the product, including the direct labor expenses, overhead expenses, travel expenses, computer use costs, etc.
3. Effort/staffing and cost estimates are based on past experience.
 - ☐ Similar projects in similar environments should be used when possible.
 - ☐ A reasonable time phasing of tasks is derived.

(Activity 10)

- ☐ Distributions of the effort/staffing and cost estimates by life-cycle stage, by task, and spread over time are prepared.
 - ☐ The effort/staffing distribution accounts for personnel turnover and specifies the number of staff by skill and the time phasing for applying and removing the staff.
4. Estimates and the assumptions made in deriving the estimates are documented, reviewed, and agreed to.

Examples of people who review and agree to the estimates and assumptions include:

- the software task leaders,
- the first-line software managers,
- the mid-level software managers,
- the project software manager,
- the system test manager, and
- the project manager.

Activity 11

Estimates for critical target computer resources are derived according to a documented procedure.

This procedure requires that:

1. Critical computer resources for the project are identified.

Examples of critical computer resources include:

- computer memory capacity,
- computer process use, and
- communications channel capacity.

(Activity 11)

2. Estimates for the critical computer resources are related to the estimates of:
 - ☐ the software product size,
 - ☐ the operational processing load, and
 - ☐ the communications traffic.
3. Estimates of the critical computer resources are documented, reviewed, and agreed to.

Examples of groups and individuals who review and agree to the estimates include:

- the software task leaders,
- the first-line software managers,
- the mid-level software managers,
- the project software manager,
- the project manager,
- the system test manager, and
- the system engineering group.

Activity 12

The project's software schedule is derived according to a documented procedure.

This procedure requires that:

1. The software schedule is related to the estimates for software size and development resources and costs.
2. The software schedule is based on past experience.
3. The software schedule accommodates the imposed milestone dates, critical dependency availability dates, and other constraints.

(Activity 12)

4. The software schedule activities and milestones are of appropriate duration/time separation to support reasonable accuracy in progress measurement.
5. The completion of software schedule activities and milestones can be objectively determined.
6. Assumptions made in deriving the schedule are documented.
7. The software schedule is documented, reviewed, and agreed to.

Examples of groups and individuals who review and agree to the software schedule include:

- the software task leaders,
- the first-line software managers,
- the mid-level software managers,
- the project software manager,
- the project manager,
- the system test manager, and
- the system engineering group.

Activity 13

The software technical, cost, resource, and schedule risks are identified, assessed, and documented.

1. The risks are analyzed and prioritized based on their potential impact to the project.
2. Contingencies for the risks are identified.

Activity 14

Plans for the project's software engineering facilities, environments, and support tools are prepared.

(Activity 14)

1. Estimates of capacity requirements for these facilities, environments, and tools are based on the project's software size estimates and other characteristics.

Examples of software development facilities, environments, and tools include:

- software development computers and peripherals,
- software test computers and peripherals,
- software engineering environment software,
- target computer environment software, and
- other support software.

2. Responsibilities are assigned and commitments are negotiated to procure or develop these facilities, environments, and tools.
3. The plans are reviewed and agreed to by all affected groups.

Examples of affected project groups include:

- software engineering (including all subgroups, such as software requirements analysis),
- system test,
- software facility maintenance, and
- configuration management.

Activity 15

Software planning data are recorded for use by the project.

1. Information recorded includes the estimates and the associated information needed to reconstruct the estimates and assess their reasonableness.
2. The software planning data are maintained under configuration management.

(Activity 15)

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Monitoring implementation

Monitor 1

Measurements are made and used to determine the cost and schedule status of the software planning activities.

1. Completions of milestones for the software planning activities are recorded and compared to the plan; status and deviations that require management action are reported to the appropriate managers.
2. Work completed, effort expended, and funds expended in the software planning activities are tracked and compared to the plan; deviations that require management action are reported to the appropriate managers.

Verifying implementation

Verification 1

The project software manager conducts regular status/coordination reviews with the project manager during the software planning activities.

1. All appropriate project technical, financial, and support groups are represented.

(Verification 1)

Examples of project groups include:

- software engineering,
- system engineering,
- hardware engineering,
- system test,
- software quality assurance,
- configuration management,
- finance,
- purchasing, and
- documentation support.

2. Status and current results of the software planning activities are reviewed against the software project's statement of work and allocated requirements.
3. Dependencies between groups are addressed.
4. Conflicts and issues not resolvable at lower levels are addressed.
5. Software risks are reviewed.
6. Action items are assigned and reviewed.
7. A summary status report from each meeting is prepared and distributed to the review participants and to other affected groups and individuals.

Verification 2

The software quality assurance group reviews and audits the activities and products for software planning, and reports results as appropriate.

At a minimum, the reviews and audits cover:

1. The process for software estimating and planning.

- (Verification 2)**
2. The process for preparing the software development plan.
 3. The content of the software development plan.
 4. The standards used for the software development plan.
 5. The process for reviewing and making project commitments.

Refer to the software quality assurance key process area for practices covering software quality assurance reviews and audits.

Software Project Tracking and Oversight

a key process area for Level 2: Repeatable

Software project tracking and oversight involves tracking and reviewing the software accomplishments and results against documented estimates, commitments, and plans, and adjusting these based on the actual accomplishments and results. A documented plan for the software effort is used as the basis for tracking the software activities, communicating status, and revising plans. The software activities are monitored by the software managers on a regular basis. Regular technical reviews and reviews with the project manager and senior management are conducted to ensure that management and staff are aware of the software status and plans, and that issues receive appropriate attention.

Goals

Goal 1	Actual results and performance of the software project are tracked against documented and approved plans.
Goal 2	Corrective actions are taken when the actual results and performance of the software project deviate significantly from the plans.
Goal 3	Changes to software commitments are understood and agreed to by all affected groups and individuals.

Commitment to perform

Commitment 1 **A project software manager is designated to be responsible for the project's software activities and results.**

Commitment 2 **The organization follows a written policy for managing a software project.**

This policy requires that:

1. A documented software development plan is used and maintained as the basis for tracking the software project.
2. Software work activities are tracked to explicitly authorized work tasks.
3. The project manager is provided with appropriate visibility of software issues.
4. Corrective actions are taken when the software plan is not being achieved, either by adjusting performance or by adjusting the software development plan.
5. Changes to the software development plan are made with the involvement and agreement of the affected groups.

(Commitment 2)

Examples of affected groups include:

- software engineering (including all subgroups, such as software requirements analysis, as well as the software task leaders),
- system engineering,
- system test,
- software quality assurance,
- software configuration management, and
- documentation support.

6. Senior management and the customer review and approve all customer commitments and commitment changes.

Ability to perform

Ability 1

A statement of work for the project is documented, approved, and controlled.

Refer to the software project planning key process area for practices covering the statement of work.

Ability 2

The system requirements allocated to software are documented, approved, and controlled.

(Ability 2)

The system requirements allocated to software are referred to as "allocated requirements" in these practices.

The allocated requirements are the subset of the system requirements that are to be implemented in the software components of the system. The elaboration and refinement of the allocated requirements result in the software requirements specification and are a primary input to the software development plan.

Refer to the requirements management key process area for practices covering the allocated requirements.

Ability 3

A software development plan for the project is documented, approved, and controlled.

Refer to the software project planning key process area for practices covering the contents and initial preparation of the software development plan.

Ability 4

The project software manager explicitly assigns management responsibilities for all major software work tasks to the mid-level software managers and first-line software managers.

The assigned responsibilities cover:

1. The work to be performed.
2. The resources.
3. The schedule.

(Ability 4)

4. The budget for these work tasks.

Ability 5

Adequate resources and budget for tracking the software project are provided.

1. The software managers and the software task leaders are assigned specific responsibilities for tracking the software project.
2. Appropriate tools to support software tracking are made available to the software managers and software task leaders.

Examples of software management and tracking tools include spreadsheet programs and project planning/scheduling programs.

Ability 6

The software managers receive required training in managing the technical and personnel aspects of the project.

Training is provided in:

1. Managing technical projects.
2. Estimating and tracking software cost and schedule.
3. Managing people.

Ability 7

First-line software managers are knowledgeable in the technical aspects of the software project.

The managers receive orientation in:

1. The project's software engineering process and procedures.
2. The project's software application domain.

Activities performed

Activity 1

A documented software development plan is used for tracking the software activities and communicating status.

This software development plan is:

1. Readily available to:

- ☐ the software engineering staff,
- ☐ the software managers,
- ☐ the project manager,
- ☐ the senior managers, and
- ☐ the other affected groups.

Examples of affected groups include:

- software quality assurance,
- system test,
- system engineering,
- software configuration management, and
- finance.

2. Regularly updated as the work progresses to reflect progress, incorporate plan refinements, and incorporate plan changes, particularly when major milestones are completed and whenever plans change significantly.
3. Updated to incorporate all commitments and changes to commitments arrived at according to a documented commitment review procedure.
4. Reviewed and approved at each revision.

(Activity 1)

Examples of people who review and approve the software development plan include:

- the software task leaders,
- the first-line software managers,
- the mid-level software managers,
- the project software manager, and
- the project manager.

5. Maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Activity 2

Senior management reviews and approves all commitments and commitment changes made to individuals and groups external to the organization.

Activity 3

Approved changes to software commitments or commitments affecting the software activities are explicitly communicated to the staff and managers of the software engineering group and software-related groups.

Activity 4

The project's software size is tracked and corrective actions are taken.

(Activity 4)

1. Sizes for all major software products and software activities are tracked.

Examples of types of software products and activities tracked for size include:

- operational software and support software,
- deliverable and nondeliverable products,
- software and nonsoftware products (e.g., documents), and
- product development and product verification activities (e.g., executing test cases).

2. Actual size of code (generated, fully tested, and delivered) is compared to the software development plan.

Examples of units of software size include source lines of code and function points.

3. Actual units of delivered documentation are compared to the software development plan.

Examples of units of documentation include:

- number of pages,
- number of requirements, and
- number of design units.

4. Overall projected software size (estimates combined with actuals) is refined, monitored, and adjusted on a regular basis.
5. Changes in software size estimates that affect commitments are resolved according to a documented commitment review procedure.

Activity 5 **The project's software costs are tracked and corrective actions are taken.**

1. Actual expenditures over time and against work completed are compared to the software development plan to identify potential overruns and underruns.
2. Software costs (per estimated element) are tracked and compared to the software development plan.
3. Effort and staffing are compared to the software development plan.
4. Changes of staffing and other software cost profiles for projected activities are resolved according to a documented commitment review procedure.

Activity 6 **The project's critical target computer resources are tracked and corrective actions are taken.**

1. The estimate and use of the project's critical computer resources are tracked for each major software component as appropriate.

Examples of critical computer resources include:

- computer memory capacity,
- computer process use, and
- communications channel capacity.

2. Changes in estimates of critical computer resources that affect commitments are resolved according to a documented commitment review procedure.

Activity 7 **The project's software schedule is tracked and corrective actions are taken.**

(Activity 7)

1. Software size and software cost measurements are adjusted to reflect schedule adjustments.
2. Software units designed, coded, unit tested, and integrated (including testing) into the next higher level component are compared to the software development plan.
3. Completion dates for test case/procedure executions and the number of executions completed are compared to the software development plan.
4. Actual completion of software activities, milestones, and other commitments is compared against the software development plan.
5. Effects of late and early completion of software activities, milestones, and other commitments are evaluated for impacts on future activities and milestones.
6. Software schedule revisions that affect commitments are resolved according to a documented commitment review procedure.

Activity 8**Software engineering technical activities are tracked and corrective actions are taken.**

1. Technical status is reported by members of the software engineering staff to their first-line manager on a regular basis.
2. System release contents for successive builds are compared to the software development plan.
3. Problems identified in any of the software engineering products are reported and documented.
4. Problems and problem fixes are tracked.

Activity 9

The software technical, cost, resource, and schedule risks are tracked throughout the life of the project.

1. The priorities of the risks and the contingencies for the risks are adjusted as additional information becomes available.
2. High-risk areas are reviewed with the project manager on a regular basis.

Activity 10

Actual measured data and replanning data for the software project tracking activities are recorded for use by the software engineering staff and managers.

1. Information recorded includes the estimates and associated information needed to reconstruct the estimates and verify their reasonableness.
2. The software replanning data are maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Activity 11

The software engineering staff and managers conduct regular reviews to track technical progress, plans, performance, and issues against the software development plan.

These reviews are conducted between:

1. The first-line software managers and their software task leaders.

(Activity 11) 2. The project software manager, mid-level software managers, and first-line software managers.

3. The project software manager and project manager.

Activity 12 **Formal reviews, to address the accomplishments and results of project software engineering, are conducted at selected project milestones and at the beginning and completion of selected stages.**

These reviews:

1. Are planned to occur at meaningful points in the project's schedule.
2. Are conducted either with the customer or internal to the organization as appropriate.
3. Use materials that are reviewed and approved by the responsible software managers where the review establishes completion of key project milestones or stages.

Examples of key milestones include software requirements baseline approval and delivery of a major set of capabilities to the customer.

4. Address the commitments, plans, and status of the software engineering activities.
5. Address the process implementations used in the software engineering and software management activities.
6. Result in the identification and documentation of significant issues, action items, and decisions.

(Activity 12)

7. Address the software risks.
8. Result in the refinement of the software development plan, as appropriate.

Monitoring implementation

Monitor 1

Measurements are made and used to determine the software cost and schedule status.

1. Completions of schedule milestones are compared to the software development plan; status and deviations that require management action are reported to the appropriate managers.
2. Software costs (per estimated element) are tracked and compared to the software development plan; status and deviations that require management action are reported to the appropriate managers.
3. Work completed, effort expended, and funds expended are compared to the software development plan; status and deviations that require management action are reported to the appropriate managers.

Verifying implementation

Verification 1

The software engineering group conducts regular status/coordination reviews with the project manager.

1. All appropriate project technical, financial, and support groups are represented.

(Verification 1)

Examples of project groups include:

- software engineering,
- system engineering,
- hardware engineering,
- system test,
- software quality assurance,
- software configuration management,
- finance,
- purchasing, and
- documentation support.

2. The technical, cost, staffing, and schedule performance is reviewed against the software development plan.
3. Computer resources designated as critical for the project are reviewed, and current estimates and actual use of these computer resources are reported against the original estimated growth profile.
4. Dependencies between groups are addressed.
5. Conflicts and issues not resolvable at lower levels are addressed.
6. Software risks are addressed.
7. Action items are assigned and reviewed.
8. A summary status report from each meeting is prepared and distributed to the review participants and to other affected groups and individuals.

Verification 2

Project management, including software engineering managers and the project manager, conducts regular oversight reviews with senior management.

- (Verification 2)**
1. The technical, cost, staffing, and schedule performance is reviewed against the software development plan.
 2. Conflicts and issues not resolvable at lower levels are addressed.
 3. Software risks are addressed.
 4. Action items are assigned and reviewed.
 5. A summary status report from each meeting is prepared and distributed to the review participants and to other affected groups and individuals.

Verification 3 **The software quality assurance group reviews and audits the activities and products for software tracking and oversight, and reports results as appropriate.**

At a minimum, the reviews and audits cover:

1. The process for reviewing and making commitments.
2. The processes for revising the software development plan.
3. The content of the software development plan.
4. The processes for tracking the project's cost, schedule, risks, and technical activities.
5. The planned reviews.

Refer to the software quality assurance key process area for practices covering software quality assurance reviews and audits.

Software Subcontract Management

a key process area for Level 2: Repeatable

Software subcontract management involves selecting a software subcontractor, establishing commitments with the subcontractor on the work to be performed, coordinating activities with the subcontractor, and tracking and reviewing the subcontractor's performance and results. The subcontractor is selected based on its ability to perform the work. A documented agreement covering the technical and nontechnical (e.g., legal, financial, and administrative) requirements is established and is the basis for managing the subcontract. Regular technical and management reviews are conducted to ensure that management and staff of both organizations are aware of the software status and plans, and that issues receive appropriate attention.

Goals

Goal 1	The prime contractor selects qualified subcontractors.
Goal 2	The software standards, procedures, and product requirements for the subcontract comply with the prime contractor's commitments.
Goal 3	Commitments between the prime contractor and subcontractor are understood and agreed to by both parties.
Goal 4	The prime contractor tracks the subcontractor's actual results and performance against the commitments.

Commitment to perform

Commitment 1 **The organization follows a written policy requiring projects to use documented standards, procedures, and processes in selecting software subcontractors and managing the software subcontract.**

Commitment 2 **A manager is designated to be responsible for establishing and managing the software subcontract.**

1. The manager is knowledgeable and experienced in software engineering or has experienced staff members assigned who have that knowledge and experience.
2. The manager coordinates the definition of the scope of work to be subcontracted, and the terms and conditions of the subcontract.
 - ☐ The project's system engineering group and software engineering group define the technical scope of the work to be subcontracted.
 - ☐ The appropriate business function groups, such as purchasing, finance, and legal, establish and monitor the terms and conditions of the subcontract.
 - ☐ The manager is responsible for selecting the software subcontractor, managing that subcontract, and arranging for the post-subcontract support of the subcontracted products.

Ability to perform

Ability 1 **Adequate resources and budget for selecting the subcontractor and managing the subcontract are provided.**

1. Both software engineering staff and managers are assigned specific responsibilities for managing the subcontract.

(Ability 1)

2. Appropriate tools for managing the subcontract are made available to the individuals managing the software subcontract.

Examples of tools for managing the subcontract include:

- estimating models,
- spreadsheet programs, and
- project management programs.

Ability 2

Software managers and other individuals establishing and managing the software subcontract are trained to perform these activities.

Training is provided in:

1. Preparing and planning for software subcontracting.
2. Evaluating a subcontract bidder's software process capability.
3. Evaluating a subcontract bidder's software estimates and plans.
4. Selecting a subcontractor.
5. Managing a subcontract.

Ability 3

The individuals managing the software subcontract receive orientation in the application domain and the software technology, environment, methodology, standards, and procedures of the subcontract.

Activities performed

Activity 1

The work to be subcontracted is defined and planned according to a documented procedure.

This procedure requires that:

1. The work to be subcontracted is selected based on a balanced assessment of both technical and nontechnical project characteristics.
 - ☐ The functions or subsystems to be subcontracted are selected to match the special skills and capabilities of potential subcontractors.
 - ☐ The specification of the work to be subcontracted is determined based on a systematic analysis and appropriate partitioning of the system and software requirements (e.g., the functions and interfaces).
2. The specification of the work to be subcontracted and the standards and procedures to be followed are derived from the project's:
 - ☐ software requirements,
 - ☐ software development plan, and
 - ☐ software standards and procedures.
3. A subcontract statement of work is:
 - ☐ prepared,
 - ☐ reviewed,
 - ☐ agreed to, and
 - ☐ maintained under configuration management.

(Activity 1)

Examples of people who review and agree to the subcontract statement of work include:

- the project manager,
- the project software manager,
- the responsible mid-level software managers,
- the responsible first-level software managers,
- the responsible software task leaders,
- the software configuration manager,
- the software quality assurance manager, and
- other appropriate contracting and business function group managers.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

4. A subcontractor selection plan is prepared concurrent with the statement of work and is reviewed and agreed to.

Examples of people who review and agree to the subcontractor selection plan include:

- the project manager,
- the project software manager,
- the responsible mid-level software managers,
- the responsible first-level software managers,
- the software configuration manager,
- the software quality assurance manager, and
- other appropriate contracting and business function group managers.

Activity 2

The software subcontractor is selected according to a documented procedure, based on a complete evaluation of the subcontract bidders' ability to perform the work.

The evaluation covers:

1. Proposals submitted for the planned subcontract.
2. Prior performance records on similar work, if available.
3. The geographic locations of the subcontract bidders' organizations relative to the prime contractor.

Effective management of some subcontracts may require frequent face-to-face interactions.

4. Software engineering and software management capabilities.

Examples of methods to evaluate subcontractors' capabilities include the SEI's Software Capability Evaluation and Software Process Assessment methods.

5. Staff available to perform the work.
6. Staffs' prior experience in similar applications, including software expertise on their software management teams.
7. Available resources.

(Activity 2)

Examples of resources include:

- facilities,
- staff,
- hardware,
- software, and
- training program.

Activity 3

The contractual agreement between the prime contractor and the subcontractor establishes the basis for managing the subcontract.

The contractual agreement includes:

1. Documented contract terms and conditions.
2. Documented statement of work that covers (directly or by reference):
 - ☐ scope of work,
 - ☐ technical goals and objectives,
 - ☐ imposed standards and procedures,
 - ☐ dependencies between the prime contractor and the subcontractor,
 - ☐ assigned responsibilities,
 - ☐ cost and schedule constraints and goals, and
 - ☐ other development constraints and goals.
3. Documented requirements specification for the products to be developed.
4. Documented software standards and procedures.

(Activity 3)

Examples of software standards and procedures include standards and procedures for:

- software development planning,
- software configuration management,
- software quality assurance,
- software design,
- problem tracking and resolution, and
- measurements.

5. Documented list of subcontractor dependencies on the prime contractor.
6. Documented subcontract data requirements list specifying the contracted deliverables.
7. Documented estimates for:
 - ☐ software size,
 - ☐ software cost,
 - ☐ software schedule, and
 - ☐ use of critical computer resources.
8. Documented acceptance procedures and acceptance criteria to be used in evaluating the subcontracted products before they are accepted by the prime contractor.

Activity 4

A documented subcontractor's software development plan, which covers (directly or by reference) the appropriate items from the prime contractor's software development plan, is reviewed and approved by the prime contractor.

(Activity 4)

Refer to the software project planning key process area for practices covering the contents of the prime contractor's software development plan.

Activity 5 A documented and approved subcontractor's software development plan is used for tracking the software activities and communicating status.

Activity 6 Changes to the subcontracted scope of work, subcontract terms and conditions, and other commitments are resolved according to a documented commitment review procedure involving affected groups of both the prime contractor and the subcontractor.

Activity 7 The prime contractor's management conducts regular status/coordination reviews with the subcontractor's management.

1. The subcontractor is provided with appropriate visibility of the needs and desires of the product's end users and customer.
2. The subcontractor's technical, cost, staffing, and schedule performance is reviewed against the subcontractor's software development plan.
3. Computer resources designated as critical for the project are reviewed; the subcontractor's contribution to the current estimates and actual use of these computer resources are reported against the original estimated growth profile for the subcontract.

(Activity 7)

4. Critical dependencies and commitments between groups and between the prime contractor and subcontractor are addressed.
 - ☐ Subcontractor commitments to the prime contractor and prime contractor commitments to the subcontractor are both reviewed.
5. Nonconformance to the subcontract is addressed.
6. Project risks involving the subcontractor's work are addressed.
7. Conflicts and issues not resolvable internally by the subcontractor are addressed.
8. Action items are assigned and reviewed.

Activity 8

Periodic technical reviews and interchanges are held with the subcontractor.

These reviews:

1. Provide the subcontractor with appropriate visibility into the customer's and end-users' needs and desires.
2. Monitor the technical activities.
3. Ensure that the subcontractor's interpretation and implementation of the technical requirements conform to the project requirements.
4. Ensure that commitments are being met and will be met in the future.
5. Ensure that technical issues are resolved in a timely manner.

Activity 9

Formal reviews to address the subcontractor's software engineering accomplishments and results are conducted at selected milestones and at the completion of selected stages.

These reviews:

1. Are pre-planned and documented in the statement of work.
2. Address the subcontractor's commitments for, plans for, and status of the software development activities and the corresponding process implementations.
3. Result in the identification and documentation of significant issues, action items, and decisions.
4. Address the software risks.
5. Result in the refinement of the subcontractor's software development plan, as appropriate.

Activity 10

The prime contractor's software quality assurance group monitors the subcontractor's software quality assurance activities according to a documented procedure.

This procedure requires that:

1. The subcontractor's resources, procedures, and standards for software quality assurance are periodically reviewed to ensure they are adequate to monitor the subcontractor's performance.
2. Regular reviews of the subcontractor are conducted to ensure the approved process is being followed.
 - The prime contractor's software quality assurance group spot checks the subcontractor's software engineering activities and products and/or audits the subcontractor's software quality assurance records, as appropriate.

(Activity 10)

3. The subcontractor's software baseline and records of its software quality assurance activities are periodically audited to assess how well the software quality assurance standards and procedures are being followed.

Activity 11

The prime contractor's software configuration management group monitors the subcontractor's activities for software configuration management according to a documented procedure.

This procedure requires that:

1. The subcontractor's resources, procedures, and standards for software configuration management are periodically reviewed to ensure they are adequate.
2. The prime contractor and the subcontractor coordinate their activities on matters relating to software configuration management to ensure that the subcontractor's products are readily integrated or incorporated into the project environment of the prime contractor.
3. The subcontractor's software baseline and records of its activities for software configuration management are periodically audited to assess how well the standards and procedures for software configuration management are being followed and how effective they are in managing the software baseline.

Activity 12

The prime contractor conducts acceptance testing as part of the delivery of the subcontractor's products according to a documented procedure.

This procedure requires that:

1. The acceptance procedures and acceptance criteria for each product are defined, reviewed, and approved by both the prime contractor and the subcontractor prior to the test.

(Activity 12)

2. The results of the acceptance tests are documented.
3. An action plan is established for any product that does not pass its acceptance test.

Activity 13

The subcontractor's performance is evaluated on a periodic basis and the evaluation is reviewed with the subcontractor.

Monitoring implementation

Monitor 1

Measurements are made and used to determine the cost and schedule status of the activities for managing the software subcontract.

1. Costs of the activities for managing the software subcontract are tracked and compared to the software development plan; status and deviations that require management action are reported to the appropriate managers.
2. Delivery dates of items of the subcontract data requirements list are recorded and compared to the plan; status and deviations that require management action are reported to the appropriate managers.
3. Delivery dates of subcontractor dependencies provided by the prime contractor are recorded and compared to the plan; status and deviations that require management action are reported to the appropriate managers.

Verifying implementation

Verification 1 **The activities for managing the software subcontract are reviewed with senior management on a regular basis.**

Refer to the software project tracking and oversight key process area for practices covering the senior management oversight reviews.

Verification 2 **The activities for managing the software subcontract are reviewed with the project manager on a regular basis.**

Refer to the software project tracking and oversight key process area for practices covering the project manager status/coordination reviews.

Verification 3 **The software quality assurance group reviews and audits the activities and products for managing the software subcontract, and reports results as appropriate.**

At a minimum, the reviews and audits cover:

1. The process for selecting the subcontractor.
2. The process for managing the software subcontract.
3. The process for coordinating software configuration management activities of the prime contractor and subcontractor.
4. The documented commitment review procedure used with the subcontractor.

- (Verification 3)**
5. The conduct of planned reviews with the subcontractor.
 6. The conduct of reviews that establish completion of key project milestones or stages for the subcontract.
 7. The acceptance process for the subcontractor's products.

Refer to the software quality assurance key process area for practices covering the software quality assurance reviews and audits.

Verification 4 **The project manager independently reviews or audits the records and/or performance of the prime contractor's software quality assurance group to ensure they are monitoring the subcontract effectively.**

Software Quality Assurance

a key process area for Level 2: Repeatable

Software quality assurance involves reviewing and auditing the software products and activities to ensure that they comply with the applicable processes, standards, and procedures, and providing the staff and managers with the results of their reviews and audits. The software quality assurance function is required on all projects. The group performing this function is independent of the software groups and project management. A senior manager who is committed to handling all major software quality assurance issues is identified. Where compliance issues exist, the software quality assurance group works with the appropriate managers, including senior management where required, to resolve the issues.

Goals

Goal 1	Compliance of the software product and software process with applicable standards, procedures, and product requirements is independently confirmed.
Goal 2	When there are compliance problems, management is aware of them.
Goal 3	Senior management addresses noncompliance issues.

Commitment to perform

Commitment 1 **The organization follows a written policy for implementing software quality assurance (SQA).**

This policy requires that:

1. The SQA function is in place on all software projects.
2. The SQA group has a reporting channel to senior management that is independent of:
 - ☐ the software engineering groups,
 - ☐ the software-related groups (e.g., software configuration management), and
 - ☐ the project managers.
3. Senior management periodically reviews the SQA activities and findings.

Ability to perform

Ability 1 **Adequate resources and budget for performing the SQA activities are provided.**

1. A manager is assigned specific responsibilities for SQA.
2. A senior manager who is knowledgeable in the quality assurance role and has the authority to take appropriate oversight actions is designated to receive and act on software noncompliance items.
 - ☐ All first-level managers and mid-level managers in the SQA reporting chain to the senior manager are knowledgeable in the quality assurance role, responsibilities, and authority.

(Ability 1)

3. Appropriate tools to support the SQA activities are made available to the SQA staff.

Examples of support tools include:

- workstations,
- database programs,
- spreadsheet programs, and
- auditing tools.

Ability 2

Members of the SQA staff are trained to perform their SQA activities.

Training is provided in:

1. Software engineering skills and practices.
2. Standards, procedures, and methods for the software project.
3. Application domain of the project.
4. Quality assurance objectives, procedures, and methods.
5. Involvement of the SQA group in the software engineering activities.
6. Effective use of SQA methods and tools.
7. Interpersonal communications.

Ability 3

The staff and managers involved in the software process receive orientation on the role, responsibilities, and authority of the SQA group in fulfilling the terms of the software project commitments.

Activities performed

Activity 1

A SQA plan is prepared for each software project according to a documented procedure.

This procedure requires that:

1. The plan is developed by the SQA group in conjunction with the project's software managers and software task leaders.
2. The plan is reviewed and agreed to.

Examples of groups and individuals who review and agree to the SQA plan include:

- the project software manager,
- the project mid-level software managers,
- the project first-level software managers,
- the SQA group,
- the system test group,
- the project manager,
- the customer SQA representative, and
- the senior manager to whom the SQA group reports noncompliance issues.

3. The plan is maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Activity 2

The SQA activities are performed in accordance with the SQA plan.

Specifically, this plan covers:

1. Responsibilities and authority of the SQA group.
2. Resource requirements for the SQA group (including staff, tools, and facilities).
3. SQA group's participation in establishing the plan and process baseline for the project.
4. Product and process evaluations to be performed by the SQA group.

Examples of products and processes to be evaluated include appropriate:

- operational software and support software,
- deliverable and nondeliverable products,
- software and nonsoftware products (e.g., documents),
- product development and product verification activities (e.g., executing test cases), and
- both the products and the processes followed in creating the product.

5. Audits and reviews to be conducted by the SQA group.
6. Project standards and procedures used as the basis for the SQA group's reviews and audits.
7. Documentation SQA is required to produce.
8. Method and frequency of providing feedback to the software engineering group and software-related groups on SQA audits and reviews.

Activity 3 **The SQA group participates in the preparation, review, and approval of the project's software development plan, process specifications, standards, and procedures.**

The SQA group ensures that appropriate and usable plans, processes, standards, and procedures are in place and can be audited.

Activity 4 **The SQA group reviews and audits the software engineering activities to ensure process compliance.**

1. The activities are evaluated against the software development plan and the designated software standards and procedures.

Refer to all other key process areas for practices covering the specific review and audit tasks performed by the SQA group.

2. Deviations are identified.

Activity 5 **The SQA group reviews representative samples of deliverable and designated nondeliverable software products to ensure compliance with the designated process requirements.**

1. The deliverable products are evaluated prior to delivery to the customer.
2. The products are evaluated against the designated software standards, procedures, and contractual requirements.
3. Deviations and product deficiencies are identified.

Activity 6 **The SQA group regularly reports the results of its reviews and audits to the software engineering staff and managers.**

Activity 7

Deviations identified in the software engineering activities are documented and handled according to a documented procedure.

This procedure requires that:

1. Deviations from the software development plan and the designated project standards and procedures are resolved with the appropriate software task leaders or software managers, where possible.
2. Deviations from the software development plan and the designated project standards and procedures not resolvable with the software task leaders or software managers are documented and presented to the senior manager designated to receive noncompliance items.
3. Noncompliance items presented to the senior manager are periodically reviewed until they are resolved.
4. The documentation of noncompliance items is maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Activity 8

The SQA group conducts regular reviews of its activities and findings with the customer's SQA personnel.

Monitoring implementation

Monitor 1

Measurements are made and used to determine the cost and schedule status of the SQA activities.

1. Completions of milestones for the SQA activities are recorded and compared to the plan; status and deviations that require management action are reported to the appropriate managers.
2. Work completed, effort expended, and funds expended in the SQA activities are tracked and compared to the plan; status and deviations that require management action are reported to the appropriate managers.
3. Numbers of product reviews, process reviews, and audits are tracked and compared to project plans.

Verifying implementation

Verification 1

The SQA activities are reviewed with senior management on a regular basis.

Refer to the software project tracking and oversight key process area for practices covering the senior management oversight reviews.

Verification 2

The SQA activities are reviewed with the project manager on a regular basis.

(Verification 2)

Refer to the software project tracking and oversight key process area for practices covering the project manager status/coordination reviews.

Verification 3

The activities of the SQA group are monitored by management outside the software project.

1. Management reviews and audits the SQA records and activities.
2. Corrective actions are identified and implemented, as appropriate.

Software Configuration Management

a key process area for Level 2: Repeatable

Software configuration management involves selecting project baseline items (e.g., the project description, products, and process specifications of the project), controlling these items and changes to them, and recording and reporting status and change activity for these items. Changes to these baseline items are controlled systematically using a defined change control process. The configuration (software and documentation) of a system, or of any of the controlled intermediate or support products, can be distinctly identified at any point in time.

Goals

Goal 1	Controlled and stable baselines are established for planning, managing, and building the system.
Goal 2	The integrity of the system's configuration is controlled over time.
Goal 3	The status and content of the software baselines are known.

Commitment to perform

Commitment 1	The organization follows a written policy for implementing software configuration management (SCM).

(Commitment 1) This policy requires that:

1. Responsibility for SCM for each project is explicitly assigned.
2. SCM is implemented on products throughout the project's life cycle.
3. SCM is implemented for externally-deliverable products and for appropriate products used inside the organization.
4. All projects have a repository for storing the key software engineering elements (i.e., configuration items) and the associated SCM records.
5. The software baselines and SCM activities are audited on a regular basis.

Ability to perform

Ability 1

A board having the authority for managing the software baselines (i.e., a software configuration control board - SCCB) is established.

The SCCB:

1. Authorizes the establishment of software baselines and their configuration items.
2. Represents the interests of the project manager and all groups who may be affected by changes to the software baselines.

(Ability 1)

Examples of affected groups include:

- software engineering,
- software estimating,
- system engineering,
- hardware engineering,
- system test,
- software quality assurance,
- software configuration management,
- documentation support, and
- software engineering process.

3. Reviews and authorizes changes to the software baselines.
4. Authorizes the creation of software baseline products.

Ability 2

A group that is responsible for coordinating and implementing SCM for the project (i.e., the SCM group) exists or is established.

The SCM group:

1. Creates the project's software baseline library.
2. Develops, documents, and distributes the SCM plans, standards, and procedures.
3. Manages access to the software baseline library.
4. Updates the software baselines.
5. Creates software baseline products.
6. Records SCM actions.

(Ability 2)

7. Produces and distributes SCM reports.

Ability 3

Adequate resources and budget for performing the SCM activities are provided.

1. A manager is assigned specific responsibilities for SCM.
2. Appropriate tools to support the SCM activities are made available to the SCM staff and to the software engineering staff.

Examples of support tools include:

- workstations,
- database programs, and
- configuration management tools.

Ability 4

Members of the SCM group are trained in the objectives, procedures, and methods for performing their SCM activities.

Ability 5

Members of the software engineering staff are trained to perform their SCM activities.

Training is provided in:

1. The standards, procedures, and methods to be followed for SCM performed inside the software engineering group.
2. The role, responsibilities, and authority of the SCM group in fulfilling the terms of the software project commitments.
3. Procedures the software engineering group must follow to implement formal SCM.

Activities performed

Activity 1

Different levels of SCM are implemented, as appropriate, during the project's life cycle.

1. The level of SCM is different for different products.
2. The level of SCM changes during the project's life cycle to provide a balance of control and flexibility that is most beneficial to the project.
3. The software engineering group informally manages its individual products during development.
4. The software engineering group uses informal SCM for products not under formal SCM.
5. The project uses formal SCM for control and stability of baselined items when needed to coordinate and interact between project groups and with the customer.
6. The project's software baseline library and its contents are controlled and available after the project ends.

Activity 2

A documented SCM plan exists.

The plan:

1. Covers the activities to be performed, the schedule of activities, the assigned responsibilities, and the resources required (including staff, tools, and computer facilities).
2. Covers the SCM requirements and activities to be performed by the software engineering group and other groups.

(Activity 2)

3. Is developed before the software development activities begin.
4. Is reviewed and agreed to by all affected groups and individuals.

Examples of affected groups and individuals include:

- the software task leaders,
- the software managers,
- the software estimating group,
- the system engineering group,
- the system test group, and
- the software quality assurance group.

5. Is maintained under configuration management.

Practices for configuration management are covered in this key process area.

Different levels of configuration management are implemented as appropriate.

Activity 3

A documented and approved SCM plan is used as the basis for performing the SCM activities.

Activity 4

A configuration management library system is established as a repository for the software baselines.

This library system:

1. Supports multiple control levels of SCM.
2. Provides for the storage and retrieval of configuration items and their configuration components.

(Activity 4)

3. Provides for the sharing (i.e., read-only) and transfer of configuration items and their configuration components between the software engineering groups' control and the SCM group's control and between control levels within the library.
4. Helps to enforce product standards (e.g., naming and format) of configuration items and their configuration components.
5. Provides for the storage and recovery of archive versions of configuration items and their configuration components.
6. Helps to ensure correct creation of software baseline products.
7. Provides for the storage, update, and retrieval of SCM records.

Examples of SCM records include:

- status and revision history of configuration items,
- identification of authorized revision of configuration items in process, and
- records of distribution of software baseline products.

8. Produces SCM reports.
9. Provides for the maintenance of the library structure and contents.

Examples of library maintenance functions include backup/restoring of library files and recovery from library errors.

Activity 5

The software engineering products and process specifications (i.e., configuration items) to be placed under configuration management are identified.

- (Activity 5)** 1. The configuration items are selected based on documented criteria.

Examples of project configuration items include:

- software requirements specifications,
- software designs,
- software code units,
- software test procedures,
- software system build for the software test activity,
- software system build for delivery to the customer,
- process specifications,
- specifications for standards and procedures,
- compilers, and
- other support tools.

2. The characteristics of each configuration item are specified.
3. The software baselines to which each configuration item belongs are specified.
4. The point in the software life cycle that each configuration item is placed under configuration management is defined.
5. The person responsible for each configuration item (i.e., the owner, from a configuration management point of view) is defined.

Activity 6 **A documented procedure is followed for initiating, recording, reviewing, approving, and tracking change requests and trouble reports for all configuration items.**

Activity 7 **A documented procedure is followed to control changes to configuration items.**

(Activity 7)

This procedure requires that:

1. Established controls are followed to ensure that configuration items are checked out and checked in for change in a manner that maintains the correctness and integrity of the software baseline library.
2. Reviews and/or regression tests are performed to ensure that changes have not caused unintended effects on the product.
3. Revised configuration items are audited to ensure that they are prepared according to the SCM standards and procedures.

Examples of standards include:

- naming standards,
- format standards, and
- required information standards such as version number and change history.

4. Only configuration items that are accepted by the SCCB are entered into the software baseline library.
 - ☐ Check-in procedures include steps to verify that the revisions are authorized, to create a change log, to maintain a copy of the changes, to update the software baseline library, and to archive the replaced software baseline.

Activity 8

A documented procedure is followed to create and control the release of software baseline products.

This procedure requires that:

1. The SCCB authorizes the creation of software baseline products.
2. Software baseline products, for both internal and external use, are built only from configuration items in the software baseline library.

Activity 9 **A documented procedure is followed to record the status of configuration items and change requests.**

This procedure requires that:

1. The configuration management actions are recorded in sufficient detail so that the software baselines' content and status are known and previous versions can be recovered.
2. The current status and history (i.e., changes and other actions) of the software baselines are maintained.

Activity 10 **Standard reports documenting the SCM activities and the contents of the software baseline are created and distributed to affected groups and individuals.**

These reports include:

1. SCCB meeting minutes.
2. Change request summary and status.
3. Trouble report summary and status (including fixes).
4. Summary of changes made to the software baselines.
5. Revision history of configuration items.
6. Software baseline status.
7. Findings of software baseline audits.

Activity 11 **A documented procedure is followed to prepare for, conduct, report results from, and track action from software baseline audits.**

(Activity 11)

The auditors:

1. Assess the integrity of software baselines.
2. Review the structure and facilities of the library system for configuration management.
3. Verify the completeness and correctness of the library contents.
4. Determine if the SCM process is followed.

Monitoring implementation

Monitor 1

Measurements are made and used to determine the cost and schedule status of the SCM activities.

1. Completions of milestones for the SCM activities are recorded and compared to the plan; status and deviations that require management action are reported to the appropriate managers.
2. Work completed, effort expended, and funds expended in the SCM activities are tracked and compared to the plan; status and deviations that require management action are reported to the appropriate managers.

Verifying implementation

Verification 1

The SCM activities are reviewed with senior management on a regular basis.

(Verification 1)

Refer to the software project tracking and oversight key process area for practices covering the senior management oversight reviews.

Verification 2

The SCM activities are reviewed with the project manager on a regular basis.

Refer to the software project tracking and oversight key process area for practices covering the project manager status/coordination reviews.

Verification 3

Periodic audits are performed to assess how well the SCM standards and procedures are being followed and how effective they are in managing the software baselines.

The audits may be performed by the SCM group or an independent group.

Verification 4

The software quality assurance group reviews and audits the activities and products for SCM, and reports results as appropriate.

At a minimum, the reviews and audits cover:

1. Compliance with the SCM standards and procedures by:
 - ☐ the SCM group,
 - ☐ the SCCB, and
 - ☐ the software engineering group.
2. Performance of periodic configuration management audits.

- (Verification 4)** 3. Performance of regular software baseline audits.

Refer to the software quality assurance key process area for practices covering the software quality assurance reviews and audits.

Organization Process Focus

a key process area for Level 3: Defined

Organization process focus involves developing and maintaining an understanding of the organization's software processes and coordinating the activities to specify and improve these processes. A group such as a software engineering process group acts as the focus for the software process activities in the organization. This group coordinates the projects' software process definition activities and the organization's long-term process improvement efforts.

Goals

Goal 1	Current strengths and weaknesses of the organization's software process are understood and plans are established to systematically address the weaknesses.
Goal 2	A group is established with appropriate knowledge, skills, and resources to define a standard software process for the organization.
Goal 3	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.

Commitment to perform

Commitment 1	Senior management sponsors the organization's activities for software process assessment, definition, and improvement.

(Commitment 1) Specifically, senior management:

1. Demonstrates to the organization's staff and managers its commitment to these software process activities.
2. Establishes long-term plans and commitments for funding, staffing, and other resources.
3. Establishes strategies for managing and implementing the activities for process definition and improvement.

Commitment 2 **Senior management oversees the organization's activities for software process definition and improvement.**

Specifically, senior management:

1. Ensures that the organization's standard software process supports its business goals and strategies.
2. Reviews, approves, and helps establish software process policies.
3. Advises on setting priorities for software process improvement.
4. Participates in establishing plans for software process improvement.
 - ☐ Senior management coordinates software process requirements and issues with higher-level corporate staff and managers.
 - ☐ Senior management coordinates with the organization's managers to secure the managers' and staff's support and participation.

Ability to perform

Ability 1

The organization establishes, funds, and staffs a group that focuses on activities for software process definition and improvement.

1. Where possible, this group is staffed by a core of software technical professionals who are assigned full time to the group, possibly supported by others on a part-time basis.

The most common example of this group is a software engineering process group (SEPG).

2. This group is staffed appropriately to represent the software engineering discipline and software-related disciplines.

Examples of software-related disciplines include software configuration management and software quality assurance.

3. Experienced staff members who have expertise in specialized areas are committed to support this group.

Examples of specialized areas include:

- software reuse,
- computer-aided software engineering (CASE) technology,
- measurements, and
- training course development.

4. Appropriate tools are made available to this group.

(Ability 1)

Examples of tools include statistical analysis tools and database systems.

Ability 2

Members of the group focusing on software process definition and improvement receive required training to perform their software process activities.

Training is provided in:

1. Software engineering practices.
2. Planning, managing, and controlling the software process.

Refer to the training program key process area for practices covering training.

Ability 3

The staff and managers of the groups involved in implementing the software processes receive orientation on the purpose of the activities for software process definition and improvement and their roles in those activities.

Activities performed

Activity 1

The software process is assessed periodically, and action plans are developed to address the assessment findings.

Assessments are typically conducted every 1-1/2 to 3 years.

(Activity 1)

Assessments look at all software processes used in the organization, but may do this by sampling process areas and projects.

An example of a method to assess an organization's software process capability is the SEI Software Process Assessment method.

Activity 2

The organization follows a documented and approved plan to coordinate its activities for software process definition and improvement.

This plan:

1. Defines the activities to be performed and the schedule for these activities.
2. Specifies the groups and individuals responsible for the activities.
3. Identifies the resources required, including staff and tools.
4. Is reviewed by the plan developers' peers when initially released and whenever major revisions are made.

Refer to the peer review key process area for practices covering peer reviews.

5. Is reviewed and agreed to by the organization's software managers and senior managers.

Activity 3

The organization's and projects' activities for defining, implementing, measuring, and improving the software process are coordinated at the organization level.

This coordination covers:

1. The development of the standard software process for the organization.

The development of the standard software process may be done by a single group or by development teams of process experts and users.

Refer to the organization process definition key process area for practices covering the development of the standard software process.

2. The development of the defined software process for each project.

Refer to the integrated software management key process area for practices covering the development of a project's defined software process.

Activity 4

The use of the software process database for the organization is coordinated at the organizational level.

The software process database is used to collect information on the software processes and resulting software products of the organization and the projects.

(Activity 4)

Refer to the organization process definition key process area for practices covering the software process database.

Activity 5

New technologies in limited use in the organization are monitored, evaluated, and, where appropriate, implemented in other parts of the organization.

Activity 6

Training for the organization's and projects' software process is coordinated across the organization.

1. Specifications for training on subjects related to the software process are prepared.
2. Where appropriate, training courses may be prepared and conducted by the group responsible for coordinating the organization's activities for software process definition and improvement (e.g., software engineering process group) or by the training group.

Refer to the training program key process area for practices covering training.

Activity 7

The staff and managers of the groups involved in implementing the software processes participate in, and are informed of, the organization's and projects' activities for software process definition and improvement.

(Activity 7)

Examples of means to involve these people include:

- electronic bulletin boards on process,
- process advisory and working groups,
- information exchange meetings,
- surveys, and
- informal discussions.

Monitoring implementation

Monitor 1

Measurements are made and used to quantify, evaluate, and track the status of the organization's activities for coordinating the process.

1. Work completed, effort expended, and funds expended in the organization's activities for process definition and improvement are tracked and compared to the plans for these activities; status and deviations that require management action are reported to the appropriate managers.
2. Results of each software process assessment are compared to the results and recommendations of previous assessments; the results are used to focus the current action plan for improving the software process.

Verifying implementation

Verification 1

Regular status and coordination reviews are conducted with the senior managers responsible for overseeing the organization's activities for software process definition and improvement.

(Verification 1) These reviews:

1. Cover progress and status of the activities to improve the software process.
2. Address conflicts and issues that cannot be resolved at lower levels.
3. Result in assignment and review of action items.
4. Result in a summary status report that is distributed to the meeting participants.

Organization Process Definition

a key process area for Level 3: Defined

Organization process definition involves establishing and maintaining a standard software process for the organization, along with related items, for use by the projects in establishing their software process. This standard software process provides a common process for software development and software maintenance projects. It defines the essential process steps for the projects and establishes the common basis for measuring process performance across all projects and for long-term improvement.

Goals

Goal 1	A standard software process for the organization is defined and maintained as a basis for stabilizing, analyzing, and improving the performance of the software projects.
Goal 2	Specifications of common software processes and documented process experiences from past and current projects are collected and available.

Commitment to perform

Commitment 1	The organization follows a written policy governing the definition of the organization's and projects' software processes.

(Commitment 1) This policy requires that:

1. A standard software process is defined for the organization.

An organization may have more than one standard software process if, for example, the organization develops significantly different types of applications.

2. The projects use a tailored version of the organization's standard software process.
3. Results from the projects are used to improve the organization's standard software process.

Ability to perform

Ability 1

Appropriate resources and funding for developing and maintaining the organization's software process assets are provided.

1. The development and maintenance of the organization's standard software process specification are performed or coordinated by the group responsible for coordinating the organization's software process activities (e.g., software engineering process group).

Refer to the organization process focus key process area for practices covering the group responsible for coordinating the organization's software process activities.

2. Appropriate tools are made available to the staff members who design, develop, tailor, specify, and maintain the software processes.

(Ability 1)

Examples of tools include:

- statistical analysis tools,
- database management systems, and
- process modeling tools.

Ability 2

The staff members who define and maintain the organization's software process assets receive required training to perform these software process activities.

Training is provided in:

1. Software engineering practices.
2. Planning, managing, and controlling the software process.

Refer to the training program key process area for practices covering training.

Ability 3

The task leaders of the groups implementing the software processes receive required training on why, when, and how the organization's standard software process should be tailored.

Refer to the training program key process area for practices covering training.

Ability 4

The staff and managers of the groups implementing the software process for each project receive orientation on the use and value of the organization's standard software process.

Activities performed

Activity 1

The organization develops and maintains a repository of software process assets for use by the projects.

These software process assets include:

1. The specification of a standard software process for the organization, which is composed of:
 - ☐ a software process architecture which describes and orders the software tasks and activities that are common across the organization's projects, and
 - ☐ the software process kernels that populate the software process architecture and which specify the fundamental tasks and activities used in building a project's software process.
2. The guidelines and criteria for tailoring the organization's software process architecture and software process kernels for use on a project.
3. The specification of software life-cycle models approved for use by the projects.
4. A library of software process specifications previously developed by projects in the organization.
5. A software process database for the organization containing information on the software processes and resulting software products of the organization and the projects.

Activity 2

The organization's standard software process is developed according to a documented procedure.

(Activity 2)

This procedure requires that the organization's standard software process:

1. Conforms to process and product standards imposed on the organization from a higher level of authority in the company, to the extent possible.
2. Conforms to process and product standards which are commonly imposed on the organization's projects by their customers, to the extent possible.
3. Incorporates state-of-the-practice software engineering tools and methods, to the extent possible.
4. Identifies and defines the necessary interfaces to the processes of the other project groups.
5. Is specified in sufficient detail so the measurements collected by the projects may be used to analyze, stabilize, and improve the organization's standard software process.
6. Is reviewed by the process developers' peers when initially developed and whenever significant changes or additions are made.

Refer to the peer review key process area for practices covering peer reviews.

7. Is maintained under configuration management.

(Activity 2)

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Activity 3

The specifications of the organization's software process kernels are documented according to established organization standards.

These standards require that the specifications of the software process kernels define:

1. The required procedures, practices, methods, and technologies.
2. The applicable process and product standards.
3. The responsibilities for implementing the process.
4. The required tools and resources.
5. The process dependencies and interfaces.
6. The process outputs.
7. The readiness and completion criteria.
8. The product and process data to be collected.

Activity 4

Software life-cycle models that are approved for use by the projects are identified and documented.

The software life-cycle models are:

1. Compatible with the organization's standard software process.

(Activity 4)

Examples of software life-cycle models include:

- waterfall,
- overlapping waterfall,
- spiral,
- serial build, and
- single prototype/overlapping waterfall.

2. Maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Activity 5

A library of software process specifications previously developed by projects in the organization is established and used.

1. The library contents are controlled and maintained by the group responsible for coordinating the organization's software process activities (e.g., software engineering process group).
2. The projects' defined software processes are reviewed and appropriate process specifications are selected for inclusion in this library.
3. The software process specifications are cataloged for easy access.
4. Revisions made to the software process specifications are reviewed, and the library contents are updated as appropriate.
5. The reuse of each process specification is reviewed periodically, and the results are used to maintain the library contents.

Activity 6

A software process database for the organization is established and used according to a documented procedure.

This procedure requires that:

1. The database is used to collect and make available information on the software processes and resulting software products of the organization and projects.
 - ☐ The database contains documented lessons learned.
 - ☐ The database contains quantitative metrics needed to calibrate software size estimates and software cost models.
 - ☐ The database contains quantitative metrics needed to plan new projects.
2. The data entered into the database is reviewed to ensure the integrity of the database contents.
3. The definition of the database is maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

4. User access to the database contents is controlled to ensure completeness, integrity, and accuracy of the process data.

Access is limited to those who have a need to enter, change, view, analyze, or extract data.

(Activity 6)

Sensitive data are protected and access to these data is appropriately controlled.

Activity 7

The organization's software process assets are revised according to a documented procedure.

This procedure requires that:

1. Changes derived from the following sources are documented and systematically reviewed:
 - ☐ the findings and recommendations of the software process assessments,
 - ☐ lessons learned from monitoring the organization's and projects' software process activities,
 - ☐ changes proposed by the organization's staff and managers, and
 - ☐ process and product measurement data that are analyzed and interpreted.
2. Changes to the software process and plans for retrofitting the changes into the organization's standard software process are reviewed and agreed to before they are incorporated.

Changes and the plans for retrofitting the changes are reviewed and agreed to by:

- the software task leaders,
- the first-line software managers,
- the mid-level software managers, and
- the project software managers.

3. Plans for introducing changes to the software process of ongoing projects are defined.

Activity 8

Large efforts to develop or revise the organization's software process assets are planned, and the plan is documented, reviewed, and approved.

Examples of individuals who review and approve the plan include:

- the software task leaders,
- the first-line software managers,
- the mid-level software managers, and
- the project software managers.

Activity 9

The group responsible for coordinating the organization's software process activities (e.g., software engineering process group) reviews changes to all projects' defined software processes that conflict with, or would improve, the organization's standard software process.

1. The organization's standard software process is revised as appropriate.
2. Waivers for changes that put the project's defined software process in conflict with the organization's standard software process are documented and are reviewed and approved by senior management as appropriate.

Monitoring implementation

Monitor 1

Measurements are made and used to quantify and evaluate the status, with respect to cost and schedule, of the organization's activities to define and improve its software processes.

1. Completion of the schedule milestones for process definition are tracked; status and problems that require management action are reported to the appropriate managers.
2. The costs for the process definition activities are updated on a regular basis and tracked; status and problems that require management action are reported to the appropriate managers.

Monitor 2

Measurements are made, analyzed, and used to control the organization's activities to define its software processes.

Projections of the organization's cost for process definition activities are updated on a regular basis and tracked; status and problems that require management action are reported to the appropriate managers.

Verifying implementation

Verification 1

The software quality assurance group audits the organization's activities and products for defining its process, and reports results as appropriate.

At a minimum, the reviews and audits confirm that:

1. The organization's standard software process is used as the foundation for each project's software process definition.

- (Verification 1)
2. The appropriate standards are followed in developing, documenting, and maintaining the organization's software process assets.
 3. The software process database for the organization is controlled and used appropriately.

Refer to the software quality assurance key process area for practices covering the quality assurance reviews and audits.

Training Program

a key process area for Level 3: Defined

Training program involves identifying the training needs of the organization, the projects, and the individuals and developing and procuring training courses to address these needs. The training program ensures that training needed to perform each of the organization's job functions is appropriate and is not circumvented inappropriately.

Goals

Goal 1	The staff and managers have the skills and knowledge to perform their jobs.
Goal 2	The staff and managers effectively use, or are prepared to use, the capabilities and features of the existing and planned work environment.
Goal 3	The staff and managers are provided with opportunities to improve their professional skills.

Commitment to perform

Commitment 1	The organization follows a written policy for meeting its training needs.

The policy requires that:

1. A required set of training courses is identified for each job function.

- (Commitment 1)**
2. Training is provided for all staff and managers.
 3. Training is provided to build the skill base of the organization, to fill the specific needs of the projects, and to develop the skills and advance the careers of individuals.
 4. Training courses are developed within the company or obtained from outside the company when appropriate.

External sources of training can include:

- commercially available training courses,
- academic programs,
- professional conferences, and
- seminars.

Ability to perform

Ability 1

Appropriate resources and funding for implementing the training program are provided.

1. A manager is designated as responsible for implementing the organization's training program.
2. A training group is established and staffed to fulfill the training needs of the organization.

The training staff may include full-time instructors or part-time instructors drawn from the organization; the staff may also be drawn from external sources.

3. Appropriate support tools are made available to the training staff.

(Ability 1)

Examples of support tools include:

- workstations,
- database programs, and
- packages for developing presentation materials.

4. Appropriate training facilities are made available to conduct training.

Training facilities should be separated from the students' work environment to eliminate interruptions.

Where appropriate, training is conducted in settings that closely resemble actual performance conditions and includes activities to simulate actual tasks and job environments.

Ability 2

Members of the training staff receive required training to teach their assigned courses.

1. Training is provided in:
 - ☐ instructional techniques, and
 - ☐ the subject matter being taught.
2. Refresher training is provided, as appropriate, to enhance the training staff's instructional skills and content knowledge.

The practices in this key process area govern this training.

Ability 3 **Managers receive orientation on the training program to enable them to determine appropriate training needs for their staff.**

Activities performed

Activity 1 **Each project develops and maintains a training plan specifying its training needs.**

The plan identifies:

1. The set of skills needed or desired within the project and when those skills are needed or desired.
2. The training that is required and the set of people for whom it is required.

Activity 2 **The organization's training plan is developed and revised periodically.**

The plan:

1. Uses the projects' training plans as a primary source of training requirements.
2. Identifies the set of skills needed or desired within the organization and when those skills are needed or desired.
3. Defines the resources (including staff, tools, and facilities) needed to prepare and conduct or procure the training courses.
4. Is reviewed by the plan developers' peers when initially released and whenever major revisions are made.

(Activity 2)

Refer to the peer review key process area for practices covering peer reviews.

5. Is reviewed and agreed to by the involved individuals.

Examples of individuals who review and agree to the training plan include:

- the senior managers,
- the software managers, and
- the managers of software-related groups.

6. Is maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

7. Is readily available to all interested parties.

Activity 3

The organization's training program is developed and revised periodically according to a documented procedure.

This procedure requires that the program:

1. Uses the organization's training plan as a primary source of training requirements.

(Activity 3)

2. Identifies the training courses required to establish the needed or desired skills of both the staff and managers of the software engineering group and related groups.

Examples of related groups include software quality assurance and software configuration management.

Refer to all the key process areas for specific training needs.

3. Maps the training courses to job functions as appropriate.
4. Identifies both internally available and externally available training courses and other training activities (e.g., professional conferences) that can satisfy the training needs.
5. *Evaluates the candidate training courses and other training activities (e.g., professional conferences) for quality and applicability, and factors the results of these evaluations into the training program as appropriate.*
6. Defines the schedule of training courses based on the projected dates the skills will be used and the projected number of students.
7. Specifies the procedures for selecting training candidates and for registering and attending the training courses.

Activity 4

A waiver procedure for required training is established and used to independently determine whether an individual possesses the knowledge and skills covered in the training course.

The evaluators are organizationally separate from the individual being evaluated and the individual's manager.

Activity 5

Where appropriate, training courses are developed, maintained, and conducted at the project level.

Examples of training appropriately done at the project level include:

- training in the specific application and requirements of the project,
- training in the project's software architecture, and
- other training more effectively or efficiently performed at the project level.

Activity 6

Training courses developed at the organization level are developed and maintained according to organization standards.

1. The following are specified for each course:
 - ☐ intended audience,
 - ☐ prerequisites for attending,
 - ☐ training objectives,
 - ☐ length of the course,
 - ☐ lesson plans,
 - ☐ criteria for determining the students' satisfactory completion,
 - ☐ procedures for regularly evaluating the effectiveness of the training, and
 - ☐ special considerations, such as piloting and field testing the course, needs for refresher training, and opportunities for follow-on training.
2. The materials for the training course are reviewed by the course developers' peers including:
 - ☐ instructional experts,
 - ☐ subject matter experts, and
 - ☐ representative students from the course being reviewed.

(Activity 6)

Refer to the peer review key process area for practices covering peer reviews.

3. The specifications for the training courses are maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Activity 7

Where appropriate, training sessions for individuals are tied to their job responsibilities so that on-the-job activities or other outside experiences will reinforce the training within a reasonable time after the course.

Activity 8

Records of training are maintained and used by management.

1. Records are kept of all students who successfully complete each course and of all students who successfully complete their designated required training courses.
2. Records of successfully completed training are considered in career assignments and advancements of the staff and managers.

Monitoring implementation

Monitor 1

Measurements are made and used to quantify and evaluate the status of the training activities.

(Monitor 1)

1. The actual attendance at each training course is compared to the projected attendance.
2. Progress in providing training courses is tracked against the organization's and projects' training plans.
3. The number of training waivers approved over time for each project and for each training course is tracked and reviewed with senior management.

Monitor 2

Measurements are made and used to quantify and evaluate the quality of the training program.

Written reviews of the training courses are collected from all students and are reviewed by the training group.

1. The results are used to determine the suitability of the course, and to improve its content and presentation.
2. The results are reviewed with the organization's managers.

Verifying implementation

Verification 1

The training program activities are reviewed with senior management on a regular basis.

Refer to the software project tracking and oversight key process area for practices covering the oversight reviews with senior management.

Verification 2 **The training group periodically reviews the training curriculum and course content to determine if they are effective, up-to-date, and relevant.**

Results are systematically reviewed and used to revise the organization's training plan, training program, and individual training courses.

Verification 3 **The training program is independently evaluated on a regular basis for consistency with, and relevance to, the organization's needs.**

1. The evaluation team consists of personnel from groups other than the training group and may include personnel from outside the organization.
2. The evaluations confirm that:
 - ☐ course records are properly maintained,
 - ☐ people designated as requiring specific training complete that training, and
 - ☐ the organization's training plan is followed.

Verification 4 **The training program activities and products are reviewed and audited, and results are reported as appropriate.**

At a minimum, the reviews and audits confirm that:

1. The process for developing and revising the training plan is followed.
2. The process for developing and revising a training course is followed.

Integrated Software Management

a key process area for Level 3: Defined

Integrated software management involves establishing and maintaining the project's defined software process and managing the software activities according to this defined software process and the special needs of the project. The project's defined software process integrates the management and technical processes as the basis for performing the project's activities. When project objectives are not being achieved, the managers know what actions need to be taken to correct the problem and reduce the likelihood of similar problems in the future. The organization provides support and historical data which the project uses to improve its software estimating, planning, and tracking process.

Goals

Goal 1	The planning and managing of each software project is based on the organization's standard software process.
Goal 2	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.

Commitment to perform

Commitment 1 **The organization follows a written policy requiring that the software projects be managed using the organization's standard software process.**

This policy requires that:

1. Each project specifies its defined software process based on the organization's standard software process.
2. The projects' tailoring of and deviations from the organization's standard software process are documented and approved.
 - ☐ The tailoring and deviations are reviewed and approved by senior management.

Ability to perform

Ability 1 **A software development plan documents the use of the project's defined software process.**

The software development plan documents the processes for both software engineering and software management.

Refer to the software project planning key process area for practices covering the content and initial preparation of the software development plan.

(Ability 1)

Refer to the software product engineering key process area for practices covering the software engineering processes.

Ability 2

The system requirements allocated to software are documented, approved, and controlled.

The system requirements allocated to software are referred to as "allocated requirements" in these practices.

The allocated requirements are the subset of the system requirements that are to be implemented in the software components of the system. The elaboration and refinement of the allocated requirements result in the software requirements specification and are a primary input to the software development plan.

Refer to the requirements management key process area for practices covering the allocated requirements.

Ability 3

Adequate resources and funding for managing the software project using its defined software process are provided.

Refer to the software project planning and software project tracking and oversight key process areas for practices covering the resource and funding elements for software management.

Ability 4

The software managers receive orientation on the organization's standard software process.

Ability 5

The software managers receive required training in managing the technical, administrative, and personnel aspects of the project and its defined software process.

Training is provided in:

1. Software estimating, planning, and tracking.
2. Methods and procedures for identifying and managing software risks.
3. The project's defined software process.

Refer to the software project planning and software project tracking and oversight key process areas for practices covering the training for software estimating, planning, and tracking.

Refer to the training program key process area for practices covering training.

Activities performed

Activity 1

The project's defined software process is developed by tailoring the organization's standard software process according to organizational guidelines and criteria.

These guidelines and criteria require that:

1. A software life-cycle model is selected and documented according to organizational guidelines to satisfy the contractual and operational constraints of the project.

(Activity 1)

Any of the software life-cycle models appropriate to the application and approved by the organization may be selected.

Examples of software life-cycle models include:

- waterfall,
- overlapping waterfall,
- spiral,
- serial build, and
- single prototype/overlapping waterfall.

2. Tailoring of the organization's standard software process for the project is reviewed and approved by the group responsible for coordinating the organization's software process activities (e.g., software engineering process group) and by senior management.
 - ☐ The software process specifications are derived from other existing defined software processes as appropriate.
 - ☐ Waivers for deviations from the organization's standard software process are documented and are reviewed and approved by senior management.
3. Waivers for deviations from contractual software process requirements are documented and are reviewed and approved by senior management and the project's customer.
4. The specification of the project's defined software process is maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Activity 2

Each project's defined software process is revised according to a documented procedure.

This procedure requires that:

1. Changes derived from the following are documented and systematically reviewed:
 - ☐ lessons learned from monitoring the software process activities of the organization's projects,
 - ☐ changes proposed by the project's staff and managers, and
 - ☐ process and product measurement data that are analyzed and interpreted.
2. Software process changes are reviewed and approved before they are incorporated.

Examples of individuals who review and approve the changes include:

- the project's software task leaders,
- the first-line software managers,
- the mid-level software managers, and
- the project software manager.

Activity 3

The management of the software project is based on the project's defined software process.

1. The project's defined software process includes provisions for gathering, analyzing, and reporting measurement data needed to manage the software activities.
2. The activities for software estimating, planning, tracking, and control are tied to the key tasks and activities of the project's defined software process.

(Activity 3)

3. Reviews are conducted at the beginning and end of each stage of the software life cycle.

Reviews at life-cycle stages determine if the stage readiness and completion criteria are satisfied.

Typical software life-cycle stages include:

- requirements analysis,
- design,
- coding,
- integration, and
- testing.

4. Readiness and completion criteria are established, documented, and used to authorize initiation and determine completion of major tasks.
5. The software development plans are adjusted based on a comparison of actual performance to planned performance according to a documented procedure.
- ☐ Deviations from the software development plan are addressed in a timely manner to prevent impact on commitments.
 - ☐ Procedures are defined and used to address deviations from the software development plan.
6. Documented criteria are defined to indicate when to replan the software project.

An example of a specific criterion is when actual or projected values of a planning parameter deviate from the planned values by a specified amount.

(Activity 3)

7. Technical and management lessons learned from monitoring the activities of the projects in the organization are systematically reviewed and used to estimate, plan, track, and replan the software engineering and software management activities.
8. The project is replanned to bring performance in line with plans according to a documented procedure.
9. Changes to commitments are arrived at according to a documented commitment review procedure.

Activity 4

The software project is staffed, trained, and managed to fulfill the special needs of the project and its defined software process.

1. The staffing plan addresses the project's needs for people with special skills and application domain knowledge.
2. Training needs are identified and documented to fit the specific needs of the project.

Refer to the training program key process area for practices covering training.

3. The planning, control, and tracking of the work is done by the software managers with inputs from the software engineering staff involved in each task.
4. The plans, processes, and procedures followed in making and tracking commitments and coordinating activities with other groups are adjusted to account for disparities between these groups and other potential problems.

(Activity 4)

Examples of disparities and problems include:

- differences in process maturity,
- process incompatibility, and
- various business factors.

Examples of other groups include:

- the customer,
- the end users,
- the subcontractors,
- the system engineering group,
- the hardware engineering group, and
- the finance group.

The end users referred to in these practices are the customer-designated end users or representatives of the end users.

Activity 5

The software process database for the organization is used for software planning and estimating according to a documented procedure.

This procedure requires that:

1. Appropriate software planning data, replanning data, and actual measured data on past, present, and planned software projects are stored in the database.

(Activity 5)

The database contains data on:

- size,
- cost,
- schedule,
- staffing, and
- technical activities.

Information recorded includes:

- the work task description,
- the assumptions,
- the estimates,
- the revised estimates,
- the actual measured data, and
- the associated information needed to reconstruct the estimates, assess their reasonableness, and derive estimates for new work.

Refer to the organization process definition key process area for practices covering the software process database.

2. Technical and management lessons learned are recorded.
3. The data entered into the database are reviewed to ensure the integrity of the contents.
4. The database is used as a primary source of data to plan, track, control, and replan a software project.
 - ☐ Similar projects in similar environments are used when possible.

(Activity 5)

5. User access to the database contents is controlled to ensure that the contents are used for their intended purposes.

Access is limited to those who have a need to enter, change, view, or extract data.

Sensitive data are protected and access to these data is appropriately controlled.

6. Parameter values used to derive estimates for software size, cost, schedule, and resource utilization are compared to those of other projects to evaluate their validity.
- ☐ Similarities and differences to the other projects in terms of application domain and design approach are assessed and recorded.
 - ☐ Rationales for similarities and differences between the parameter values are recorded.
 - ☐ The reasoning used to judge the credibility of project estimates is recorded.

Activity 6

The project's software size is quantitatively managed according to a documented procedure.

This procedure requires that:

1. The software size is estimated and tracked.

Refer to the software project planning and software project tracking and oversight key process areas for practices covering estimating and tracking software size.

(Activity 6)

2. A group that is independent of the software engineering group reviews the procedures for estimating software size, ensures that they are followed, and provides guidance in using historical data to establish credible estimates.

An example of an independent group is a software estimating group.

An example of a method to evaluate the credibility of software size estimates is a function-by-function comparison to a completed system.

- ☐ The individuals who prepared the software size estimates ensure that the procedures and data used in the estimates are appropriate.
 - ☐ When the validity of a software size estimate is questioned, a team of peers and experts reviews the estimate.
3. A contingency factor is applied to the software size estimate for each software element identified as a software risk.
 - ☐ The rationale for the contingency is documented.
 - ☐ The risks associated with reducing or eliminating the contingency are assessed and documented.
 4. Off-the-shelf or reusable software components are identified.
 - ☐ Reuse measurements account for the reuse of requirements, design, code, test plan/procedures, etc.
 - ☐ The effort to modify and incorporate reusable components is factored into the size estimates.
 5. Factors which significantly affect software size are identified and monitored closely.

(Activity 6)

6. Software size estimates are regularly revised as additional information becomes available.
 - ☐ As the project progresses, actual software measurements are used for components that are available and integrated, including newly developed software and off-the-shelf and reusable software components.
7. A software size threshold is established for each managed element which, when projected to be exceeded, requires action.
8. The initial estimates of software size and significant revisions are reviewed by peers of the individuals who developed the estimates.

Refer to the peer review key process area for practices covering peer reviews.

Activity 7

The project's software costs are managed against the key tasks of the defined software process according to a documented procedure.

This procedure requires that:

1. Software costs are estimated and tracked.

Refer to the software project planning and software project tracking and oversight key process areas for practices covering estimating and tracking software costs.

2. Software cost models and staffing profile models, if used, are adapted to the project and use available historical data where appropriate.

(Activity 7)

3. Referenced productivity and cost data are adjusted to incorporate project variables.

Examples of project variables include:

- the geographic locations of the project development organizations,
- the size and complexity of the system,
- the stability of the requirements,
- the host environment for development,
- the target environment of the system,
- the developers' familiarity and skill with the application,
- the availability of resources, and
- other special constraints.

4. The overall software cost is allocated to individually managed elements (e.g., tasks or stages) as needed to effectively manage the cost.
5. When the software cost status is reviewed and the estimates are revised, actual expenditures over time and against work completed are compared to the software development plan and used to refine the cost estimates for remaining work.
- ☐ The software cost model used in estimating software costs is updated whenever major changes are made to the software requirements.
 - ☐ Actual data on project productivity and other new software costs are used where appropriate.
6. A software cost threshold is established for each managed element (e.g., task or stage) which, when projected to be exceeded, requires action.
7. The initial estimates of software cost and significant revisions are reviewed by peers of the individuals who developed the estimates.

(Activity 7)

Refer to the peer review key process area for practices covering peer reviews.

Activity 8

The project's critical target computer resources are managed according to a documented procedure.

This procedure requires that:

1. The project determines which resources are critical.

Examples of critical computer resources include:

- computer memory capacity,
- computer process use, and
- communications channel capacity.

2. The project's critical computer resources are estimated and tracked.

Refer to the software project planning and software project tracking and oversight key process areas for practices covering estimating and tracking the project's critical computer resources.

3. Estimates for the project's critical computer resources are derived based on historical experience, simulations, prototyping, or analysis as appropriate.
 - ☐ Sources and rationale for estimates are documented.
 - ☐ Similarities and differences between the project and the sources for historical data in terms of application domain and design approach are assessed and recorded.
 - ☐ The reasoning used to judge the credibility of estimates is recorded.

(Activity 8)

4. The planned computer resources, the system requirements allocated to software, and the software design are adjusted to achieve the project's computer resource requirements.
5. The available critical computer resources are allocated to the major software components.
6. The available capacity for the critical computer resources provides for a specified reserve capacity when the initial estimates are made.
7. The estimates and actual use of the critical computer resources are closely tracked on a regular basis.
8. A threshold is established for each critical computer resource which, when projected to be exceeded, requires action.

In sensitive cases, an audit team of technical experts:

- reviews the situation,
- assesses the risks, and
- recommends actions.

9. Changes in estimates of critical computer resources that affect commitments are resolved according to a documented commitment review procedure.
10. The initial estimates of critical computer resources and significant revisions are reviewed by peers of the individuals who developed the estimates.

Refer to the peer review key process area for practices covering peer reviews.

Activity 9

The critical dependencies and critical paths of the project's software schedule are managed according to a documented procedure.

This procedure requires that:

1. The software schedule is planned and tracked.

Refer to the software project planning and software project tracking and oversight key process areas for practices covering planning and tracking the software schedule.

2. Milestones, tasks, commitments, critical dependencies, staffing, costs, and reviews are allocated in the schedule consistent with the project's defined software process.

Different levels of schedule detail, appropriately tied to each other, are provided to accommodate the needs of different groups and individuals.

The schedule identifies specific tasks and milestones whose completion can be objectively determined (i.e., a binary or yes/no determination).

3. Critical dependencies (external and internal) are defined, negotiated, and reflected in the software schedule.

Critical dependencies include both those within the software engineering group (i.e., between subgroups) and between the software engineering group and other groups.

(Activity 9)

Refer to the intergroup coordination key process area for practices covering the handling of critical dependencies.

4. Schedule critical paths are defined and reflected in the software schedule.
5. The project's critical dependencies and schedule critical paths are tracked on a regular basis.
6. Specific documented criteria are established for each critical path which, when satisfied, require action.
 - ☐ Analyses and simulations are conducted to trade off function, quality (especially for an interim product delivery), cost, schedule, staffing, and other resources for the critical path requiring action.
 - ☐ Unallocated contingencies and schedule slack, if available, are allocated as appropriate.
 - ☐ The effects of contemplated actions on all critical paths are evaluated.
 - ☐ Decisions are made visible to all groups and individuals who may be affected.

Activity 10

The project's software risks are identified, assessed, documented, and managed according to a documented procedure.

This procedure requires that:

1. Risks potentially affecting the software activities are identified, analyzed, and ranked, and contingencies are identified.

Risks can be in areas internal to the software engineering group or can be associated with the customer, end users, subcontractors, or other groups within the organization.

(Activity 10)

Refer to the software project planning key process area for practices for identifying risks.

2. Contingency planning is based on the project's defined software process and is performed throughout the project's life cycle.

Contingency planning covers:

- technical feasibility;
- errors in size, cost, schedule, staffing, and resource estimates;
- requirements and design changes;
- external commitments;
- omissions from initial estimates;
- management reserves; and
- variations in the averages used to estimate the work.

3. Alternatives for each risk are defined, where possible, along with criteria for selecting among the alternatives.
4. Plans to mitigate the risks are established and followed.
5. The initial release and major revisions to the software risk management plan are reviewed by the plan developers' peers.

Refer to the peer review key process area for practices covering peer reviews.

6. The risk management plan is maintained under configuration management.

(Activity 10)

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

7. Risks are tracked, reassessed, and replanned at selected project milestones and designated risk checkpoints, and during the planning of significant project changes.
 - ☐ Risk priorities and risk management plans are reviewed and revised at these reassessment points.
 - ☐ Information obtained from monitoring the risks are used to refine the risk assessments and risk management plans.
8. Project staff and managers, subcontractors, customers, and end users are included in the communications on project risks, risk management plans, and results of risk mitigation as appropriate.

Activity 11

A review of the software project is periodically performed to determine the actions needed to bring the project's performance and results in line with the current and projected business, customer, and end-user needs.

Examples of future actions include:

- accelerating the schedule,
- changing the system requirements in response to a change in market opportunities and customer and end-user needs, and
- terminating the project.

Monitoring implementation

Monitor 1

Measurements are made and used to determine the performance of the software management process and refine the plans for remaining work.

1. The size of the software is compared to the original estimate and to previously revised estimates to determine if estimates need to be revised and if costs or schedules need to be replanned.
 - ☐ The actual and estimated sizes identify and include the amount and type of software reused.
2. Projections for schedule milestones are updated on a regular basis and compared to the software development plan; status and deviations that require management action are reported to the appropriate managers.
3. Projections of costs are updated on a regular basis and compared to the software development plan; status and deviations that require management action are reported to the appropriate managers.
4. The number of defects found in each reporting period are tracked, and the results are used to adjust the staffing planned for detecting and correcting defects.

Monitor 2

Measurements are made and used to assess the quality of the software management process.

The frequency, causes, and magnitude of replanning efforts are recorded and used to evaluate the quality of the software management process.

Monitor 3

Measurements are made and used to determine the performance of the process for managing software risk.

(Monitor 3)

1. Staffing over time, personnel turnover, and the availability of key and experienced personnel are compared to the software development plan; status and deviations that require management action are reported to the appropriate managers.
2. The expected availability of elements and products upon which development activities depend is tracked and compared to the software development plan; status and deviations that require management action are reported to the appropriate managers.
3. Use of the project's critical computer resources is compared to available capacities and to previously planned growth profiles; problems are identified and analyzed to determine if:
 - ☐ committed capabilities need to be reduced,
 - ☐ more computer resources need to be added, or
 - ☐ other actions need to be taken.

Examples of critical computer resources include:

- processor use,
- memory use, and
- communications load.

Verifying implementation

Verification 1

The activities for managing the software project are reviewed with senior management on a regular basis.

Refer to the software project tracking and oversight key process area for practices covering the oversight reviews with senior management.

Verification 2 **The activities for managing the software project are reviewed with the project manager on a regular basis.**

Refer to the software project tracking and oversight key process area for practices covering the status/coordination reviews with the project manager.

Verification 3 **The software quality assurance group reviews and audits the activities and products for managing the software project, and reports results as appropriate.**

At a minimum, the reviews and audits cover:

1. The processes for developing and revising the project's defined software process.
2. The processes for managing the project's software cost, schedule, risk, and technical activities.
3. The process for using and controlling the software process database for software planning and estimating.

Refer to the software quality assurance key process area for practices covering the quality assurance reviews and audits.

Software Product Engineering

a key process area for Level 3: Defined

Software product engineering involves performing the technical activities to build and maintain the software system using appropriate state-of-the-practice tools and methods. The software requirements are identified, analyzed, refined, and documented. A software architecture and software designs are developed to implement the software requirements. The program code is developed to implement the software architecture and software designs. The software evaluation activities ensure that the software product to be delivered satisfies the specified requirements. A balance of flexibility and control is maintained to correct and revise the requirements, designs, and code to incorporate corrections and enhancements identified throughout the life cycle.

Goals

Goal 1	Software engineering issues for the product and the process are properly addressed in the system requirements and system design.
Goal 2	The software engineering activities are well-defined, integrated, and used consistently to produce a software system.
Goal 3	State-of-the-practice software engineering tools and methods are used, as appropriate, to build and maintain the software system.

Goal 4 **Software engineering products that are consistent with each other and appropriate for building and maintaining the software system are systematically developed.**

Commitment to perform

Commitment 1 **The organization follows a written policy that guides the software engineering activities.**

This policy requires that:

1. The best technical tools and methods, appropriate to the project, are used to build and maintain the software system.
2. The software engineering staff is skilled in the software engineering tools and methods used on the project.
3. The software plans, activities, and products are kept in conformance with the baseline of the system requirements allocated to software.

The system requirements allocated to software are referred to as "allocated requirements" in these practices.

The allocated requirements are the subset of the system requirements that are to be implemented in the software components of the system. The elaboration and refinement of the allocated requirements result in the software requirements specification and are a primary input to the software development plan.

Ability to perform

Ability 1

Adequate resources and funding for performing the activities of software product engineering are provided.

1. Skilled staff members are available to perform the different software engineering activities, including:
 - ☐ requirements analysis,
 - ☐ design,
 - ☐ coding, and
 - ☐ testing.
2. Appropriate tools and support needed to perform the software engineering activities are made available to the software engineers.

Examples of appropriate general tools include:

- workstations,
- database management systems,
- graphics tools,
- document processing systems, and
- word processing systems.

Examples of appropriate tools for analyzing the software requirements include tools for:

- specifying,
- prototyping,
- modeling, and
- simulating.

(Ability 1)

Examples of appropriate tools for software design include tools for:

- specifying,
- prototyping, and
- simulating.

Examples of appropriate tools for coding include:

- editors,
- compilers,
- cross-reference generators, and
- pretty printers.

Examples of appropriate tools for testing the software include:

- test management tools,
- test generators,
- test drivers,
- test profilers,
- symbolic debuggers, and
- test coverage analyzers.

Ability 2

The project manager, project software manager, and all software managers receive orientation in the software system and in the software engineering tools and methods to be used by the project.

This orientation covers:

1. The application domain.
2. Deliverable and nondeliverable software products that will be developed and maintained.

(Ability 2)

3. Guidelines on how to manage the project using the chosen methods and tools.

Ability 3

Members of the software engineering staff receive required training to perform their technical assignments.

1. The staff members involved in the software engineering process are cross-trained, as appropriate, in the tools, methods, conventions, and standards used by the project in:
 - ☐ analyzing requirements,
 - ☐ designing the software,
 - ☐ coding, and
 - ☐ testing.
2. Training is provided to software requirements analysts in:
 - ☐ the application domain,
 - ☐ principles of analyzing software requirements,
 - ☐ skills to interview end users and application domain experts in order to establish the software requirements, and
 - ☐ use of the tools, methods, conventions, and standards selected by the project for analyzing software requirements.
3. Training is provided to software designers in:
 - ☐ the application domain,
 - ☐ design concepts, and
 - ☐ use of the tools, methods, conventions, and standards selected by the project for designing software.
4. Training is provided to programmers in:
 - ☐ the application domain;
 - ☐ programming concepts;
 - ☐ the selected programming language(s);
 - ☐ use of the tools, methods, conventions, and standards selected by the project for programming;

(Ability 3)

- ☐ use of the selected programming tools, methods, conventions, and standards;
 - ☐ programming style and coding conventions; and
 - ☐ unit testing techniques.
5. Software testers may be involved in different levels of testing (e.g., unit, integration, and system). Training is provided to software testers in:
- ☐ testing concepts;
 - ☐ test planning;
 - ☐ use of the tools, methods, conventions, and standards selected by the project for testing the software;
 - ☐ criteria for test readiness and completion; and
 - ☐ measuring test coverage.

Refer to the training program key process area for practices covering training.

Activities performed

Activity 1

Appropriate state-of-the-practice software engineering tools and methods are integrated with the project's defined software process and software life cycle.

1. The software engineering activities are integrated according to the project's defined software process.

Refer to the integrated software management key process area for practices covering the project's defined software process.

2. Tools and methods appropriate for use on the project are selected.

(Activity 1)

- ☐ Candidate tools and methods are selected based on their applicability to the project's defined software process, the existing skill base, availability of training, contractual requirements, power and ease of use, support services, etc.
 - ☐ The rationale for selecting a particular tool or method is documented.
3. Configuration management models appropriate to the project are selected.

Examples of configuration management models include:

- check-out/check-in models,
- composition models,
- transaction models, and
- change set models.

Activity 2

The software engineering group actively participates in the control of the system requirements allocated to software.

The software engineering group:

1. Has free access to:
 - ☐ the baseline of the allocated requirements,
 - ☐ the rationale for allocating the system requirements to software, and
 - ☐ the descriptions of proposed and planned changes.
2. Reviews, assesses the impact of, and approves proposed changes to the allocated requirements.
3. Reviews and approves all baselines of the allocated requirements as a readiness criterion for the software requirements analysis task.

Activity 3

The software requirements are developed, documented, and verified by systematically analyzing the allocated requirements according to the project's defined software process.

1. The staff involved in analyzing the software requirements reviews the allocated requirements to ensure that issues affecting the software requirements analysis are resolved before they have an impact.
 - ☐ Software requirements cover the software functions and performance, the interfaces to both hardware and software, and the computer-human interface.
2. Effective methods for requirements analysis are used to identify and derive the software requirements.

Examples of effective methods for requirements analysis include:

- requirements elicitation,
- prototyping, and
- scenario generation.

3. Investigations, tradeoff studies, simulations, modeling, and prototyping to identify specific software requirements are performed and documented, as appropriate, along with an analysis of the alternatives and the rationale for the selected alternative.
4. The software requirements are individually analyzed to ensure they are feasible and appropriate to implement in software, clearly stated, consistent with each other, and verifiable.
 - ☐ Software requirements which are problems are identified and reviewed with the team responsible for the system requirements, as appropriate, and appropriate changes are made to the allocated requirements and the software requirements.
5. The software requirements are documented in a software requirements specification.

(Activity 3)

6. The group responsible for formal testing of the software analyzes each software requirement to ensure it can be verified.
7. The methods for verifying that each software requirement is satisfied are identified and documented.

Examples of verification methods include:

- demonstration,
- test,
- analysis, and
- inspection.

8. The software requirements specification is reviewed and approved by:
 - ☐ the project manager,
 - ☐ the system engineering manager,
 - ☐ the project software manager, and
 - ☐ the manager responsible for the formal testing of the software.
9. The software requirements specification is reviewed with the customer and end users.

The end users referred to in these practices are the customer-designated end users or representatives of the end users.

10. The software requirements specification is reviewed by the specification developers' peers before it is considered complete.

Refer to the peer review key process area for practices covering peer reviews.

(Activity 3)

11. The software requirements specification is appropriately changed whenever the allocated requirements change.

Refer to the requirements management key process area for practices covering the revision and control of the allocated requirements.

Activity 4

The software design is developed, documented, and verified, according to the project's defined software process, to accommodate the software requirements and to form the framework for coding.

1. The staff involved in software design reviews the software requirements to ensure that issues affecting the software design are resolved before they have an impact.
2. Application standards, such as operating system interfaces, computer-human interfaces, and networking interfaces, are used where appropriate.
3. Effective methods are used to design the software system.

Examples of effective software design methods include:

- structural models,
- design reuse,
- object-oriented design, and
- essential systems analysis.

4. The software architecture (i.e., the top-level software framework of the system) is developed early (within the constraints of the software life cycle and technology being used).

(Activity 4)

5. The software architecture is reviewed to ensure that architecture issues affecting the software design are resolved before they have an impact.
6. The software design is documented.
 - ☐ The documentation of the software design covers the software components; the computer-human interface; and the interfaces between software components, to other software systems, and to hardware.
7. The software design document is reviewed by the designers' peers before it is considered complete.

Refer to the peer review key process area for practices covering peer reviews.

8. The software design document is appropriately changed whenever the software requirements change.

Activity 5

The software code is developed and verified, according to the project's defined software process, to implement the software requirements and software design.

1. The staff involved in coding reviews the software requirements and software design to ensure that issues affecting the coding are resolved before they have an impact.
2. Effective programming methods are used to code the software.

Examples of effective programming methods include structured programming and code reuse.

(Activity 5)

Examples of programming style and conventions include:

- the use of language features, formatting, commenting, and naming of software units;
- unit size limits;
- complexity limits; and
- other structural parameters.

3. Critical and/or shared code modules are developed early.
4. Each code unit is reviewed by the programmers' peers before it is considered complete.

Refer to the peer review key process area for practices covering peer reviews.

5. The code units are integrated to form a software system.
6. The code is appropriately changed whenever the software requirements or software design changes.

Activity 6

Effective techniques are used to test the software at all levels of testing.

1. The adequacy of testing is determined based on:
 - ☐ the level of testing performed,
 - ☐ the test strategy selected, and
 - ☐ the test coverage to be achieved.

(Activity 6)

Examples of levels of testing include:

- unit testing,
- integration testing,
- system testing, and
- acceptance testing.

Examples of test strategies include:

- functional (black-box),
- structural (white-box), and
- statistical.

Examples of test coverage approaches include:

- statement coverage,
- path coverage,
- branch coverage, and
- usage profile.

2. For any level of testing, the software must have satisfied its readiness criteria.

Software units have successfully completed a code peer review and unit testing before they enter integration testing.

The software has successfully completed integration testing before it enters system testing.

(Activity 6)

A test readiness review is held before the software enters acceptance testing.

3. Regression testing is performed, as appropriate, at each test level whenever the software being tested changes.
4. The test plan, test procedures, and test cases are reviewed by peers of the developers of the plan and procedures before they are considered ready for use.

Refer to the peer review key process area for practices covering peer reviews.

5. Test plans, test procedures, and test cases are appropriately changed whenever the allocated requirements, software requirements, software design, or code being tested changes.

Activity 7

Formal system testing of the software is performed, according to the project's defined software process, to ensure that the software satisfies the software requirements.

1. System testing of the software is performed according to a documented software test plan that is reviewed with the customer and end users.
 - ☐ The software test plan covers types of testing to be performed, test strategies, test coverage approaches, methods and approach for tracing requirements to test cases, and reliability metrics.
2. The software test cases and test procedures are prepared by a group that is separate from the group who developed the software.
3. The test cases for system testing of the software are documented and reviewed with the customer and end users.

(Activity 7)

4. Resources for system testing of the software are assigned early enough to provide for adequate test preparation. This includes scheduling testing resources and developing test drivers and simulators.
5. System testing of the software is performed against a baselined software requirements specification and software system.

Activity 8

Acceptance testing of the software is performed, according to the project's defined software process and approved acceptance test plan, to demonstrate to the customer and end users that the software satisfies the allocated requirements.

1. Acceptance testing is documented in a software acceptance test plan, which is reviewed with and approved by the customer and end users. The acceptance test plan covers:
 - ☐ the overall testing and verification approach;
 - ☐ responsibilities of the developing organization, subcontractors, customer, and end users;
 - ☐ test facility, test equipment, and test support requirements; and
 - ☐ acceptance criteria.
2. The software acceptance test cases and test procedures are prepared by a group that is organizationally separate from the group who developed the software.
3. The test cases for software acceptance testing are documented and are reviewed with and approved by the customer and end users before the testing begins.
4. Acceptance testing of the software is performed against a baselined software system and a baselined specification of the allocated requirements.

Activity 9

The documentation that will be used to operate and maintain the system is reviewed and approved by the customer, end users, and system maintainers.

1. Preliminary versions of these documents are developed and made available early in the life cycle for the customer's, end-users', and system maintainers' review and feedback.

Examples of such documents include:

- the user's manual,
- the operator's manual, and
- the maintenance manual.

2. Final versions of these documents are verified against the software system baselined for the software acceptance testing.

Activity 10

Data on defects identified in peer reviews and system testing of the software are collected and analyzed according to a documented procedure.

This procedure requires that the following data be collected and analyzed:

1. Defect description.
2. Defect category.
3. Severity of the defect.
4. Units containing the defect.
5. Units affected by the defect.
6. Activity where the defect was introduced.

(Activity 10)

7. Test cases that identified the defect.
8. Description of the scenario being run that identified the defect.
9. Expected result and actual results that identified the defect.

Activity 11

Consistency is maintained across software engineering products, including the software plans, allocated requirements, software requirements specification, software design, code, test plans, and test procedures.

1. Software products are documented. The documentation is available to the project staff members who need it.
2. The software requirements, software design, code, and test cases for system and acceptance testing are traced to the source from which they were derived and to the products of the subsequent software engineering activities.
3. The key software engineering work products are maintained under configuration management, including:
 - ☐ software requirements specification;
 - ☐ software design document;
 - ☐ software code units;
 - ☐ software test plans, test cases, and test procedures;
 - ☐ documentation tracing the allocated requirements through the software requirements, design, code, and test cases; and
 - ☐ documentation that will be used to operate and maintain the system.

(Activity 11)

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

4. As the understanding of the system improves after each stage in the software life cycle, changes to the software work products, plans, and processes are proposed, analyzed, and incorporated as appropriate.
 - ☐ The project determines the impact of the change before the change is made.
 - ☐ Where changes to the allocated requirements are needed, they are approved and incorporated before any software work products or activities are changed.
 - ☐ Changes to all software products, plans, processes, and activities are coordinated.
 - ☐ Changes are negotiated with and communicated to the groups and individuals who may be affected.
 - ☐ Changes are tracked to completion.

Monitoring implementation

Monitor 1

Measurements are made and used to quantify and evaluate the functionality and quality of the software products.

1. The number, type, and severity of defects identified in the software products are tracked over time and by stage; status and deviations that require management action are reported to the appropriate managers.

(Monitor 1)

2. The allocated requirements are summarized by category. Within these categories the allocated requirements are traced to the software requirements and to the system test cases to assess the expansion and coverage of the allocated requirements; status and deviations (e.g., gaps in coverage and unusual numbers of requirements in categories) that require management action are reported to the appropriate managers.

Examples of categories of requirements include:

- security,
- system configuration,
- interface,
- functional,
- performance,
- usability,
- operating environment,
- reliability,
- maintainability, and
- other quality requirements.

Monitor 2

Measurements are made and used to control and stabilize the software engineering activities.

1. The status of each allocated requirement is individually tracked throughout the life of the project.

(Monitor 2)

Examples of status conditions include:

- a baseline of the allocated requirement is established,
- software requirements covering the allocated requirement are completed,
- design covering the allocated requirement is completed,
- coding covering the allocated requirement is completed,
- verification of the allocated requirement is completed,
- formal system verification of the allocated requirement is completed, and
- acceptance testing of the allocated requirement is completed.

2. Problem reports are tracked by severity and length of time they are open; status and deviations that require management action are reported to the appropriate managers.
3. Change activity for the allocated requirements is tracked; status and issues that require management action are reported to the appropriate managers. The following are tracked:
 - ☐ effort and other costs to analyze proposed changes and get agreement on their disposition, for each proposed change and total over time;
 - ☐ number of changes incorporated into the system baseline by category (e.g., interface, security, system configuration, performance, and usability); and
 - ☐ size and cost to implement and test each major incorporated change, including initial estimate and actual size and cost.

Verifying implementation

Verification 1 **The activities for software product engineering are reviewed with senior management on a regular basis.**

Refer to the software project tracking and oversight key process area for practices covering the oversight reviews with senior management.

Verification 2 **The activities for software product engineering are reviewed with the project manager on a regular basis.**

Refer to the software project tracking and oversight key process area for practices covering the status/coordination reviews with the project manager.

Verification 3 **The software quality assurance group reviews and audits the project's activities and products for software engineering, and reports results as appropriate.**

At a minimum, the reviews and audits confirm that:

1. The software requirements are reviewed to ensure that they are:
 - ☐ complete,
 - ☐ correct,
 - ☐ consistent,
 - ☐ feasible, and
 - ☐ verifiable.
2. Readiness and completion criteria for each software engineering activity are satisfied.

- (Verification 3)
3. Software products comply with the standards and requirements specified for them.
 4. Required testing is performed.
 5. Formal system testing of the software is performed according to documented plans and procedures.
 6. Acceptance testing is performed according to the approved plans and procedures.
 7. Tests satisfy their adequacy criteria.
 8. Tests are satisfactorily completed and recorded.
 9. Problems detected are documented, tracked, and addressed.
 10. Tracing of the allocated requirements through the software requirements, design, code, and test is performed.

Refer to the software quality assurance key process area for practices covering the quality assurance reviews and audits.

Intergroup Coordination

a key process area for Level 3: Defined

Intergroup coordination involves the disciplined interaction and coordination of the project groups with each other to address system-level issues and activities. System-level objectives and plans are established and used as the cornerstone for all project activities. The project groups participate, as appropriate, in defining the system requirements; establishing a system configuration; allocating requirements to hardware, software, firmware, and manual processes; monitoring and reviewing the hardware and software design and development; and managing and controlling changes to the system throughout the development effort. The technical working interfaces and interactions between groups are planned and managed to ensure the quality and integrity of the entire system.

Goals

Goal 1	The project's technical goals and objectives are understood and agreed to by its staff and managers.
Goal 2	The responsibilities assigned to each of the project groups and the working interfaces between these groups are known to all groups.
Goal 3	The project groups are appropriately involved in intergroup activities and in identifying, tracking, and addressing intergroup issues.
Goal 4	The project groups work as a team.

Commitment to perform

Commitment 1 **The organization follows a written policy requiring an environment enabling people from different disciplines to work together.**

This policy requires that:

1. The project-level objectives and the work baseline for the project are defined and agreed to by all project groups.
2. The project groups appropriately coordinate their plans and activities.
3. Managers are responsible for establishing and maintaining an environment to facilitate interaction, coordination, support, and teamwork across all groups of the project, including the customer and end users, and across the organization to ensure the success of the project and the organization.

The end users referred to in these practices are the customer-designated end users or representatives of the end users.

Ability to perform

Ability 1 **Appropriate resources and funding for coordinating the software engineering activities with the other engineering groups are provided.**

Ability 2

The support tools used by the different project groups are compatible to enable effective communication and coordination.

Examples of tools that need to be compatible include:

- word processing systems,
- database systems,
- graphics tools,
- spreadsheet programs, and
- problem tracking packages.

Ability 3

All managers in the organization receive required training in teamwork.

Training is provided in:

1. Building teams.
2. Managing teams.
3. Establishing, promoting, and facilitating teamwork.

Refer to the training program key process area for practices covering training.

Ability 4

All task leaders in the engineering groups receive orientation in the processes, methods, and standards used by the various project engineering groups.

(Ability 4)

Examples of project engineering groups include:

- software engineering,
- system engineering,
- hardware engineering,
- system test,
- software quality assurance,
- software configuration management, and
- documentation support.

Ability 5

The organization's staff receives orientation in the operating methods for teams.

Activities performed

Activity 1

The software engineering group and the other project engineering groups actively participate with the customer and end users to establish their system needs.

Specifically, these groups:

1. Operationally define the critical characteristics of the customer's and end-users' needs.
2. Document the acceptance criteria.

Activity 2

Representatives of the software engineering group work with representatives of the other project engineering groups to monitor and coordinate the project-level technical activities and resolve project-level technical issues.

(Activity 2)

The intergroup team follows a documented procedure to:

1. Coordinate the specification and provide the technical review and approval of the system requirements and system design.

The system requirements and system design are developed by the system engineering group.

The system requirements and system design include:

- the overall system requirements,
 - the system configuration (i.e., hardware, software, firmware, and human components),
 - the allocation and tracing of requirements to these system components, and
 - the definitions of the interfaces between these system components.

2. Resolve project-level conflicts and clarify system requirements and design issues.
3. Provide the project-level technical review and analysis needed to manage and control changes to the system baselines throughout the development.
4. Monitor and review the hardware, software, firmware, and manual process design and development activities.
5. Develop joint recommendations to resolve problems in order to maximize the quality and integrity of the entire project.
6. Assess, develop recommendations for, and track technical risks that involve more than one project group.

(Activity 2)

Refer to the integrated software management key process area for practices covering risk management.

7. Provide a project-wide focus on process to address process issues that span engineering disciplines.

Activity 3

A documented plan is used to communicate intergroup commitments and coordinate and track the work performed.

This plan is:

1. The baseline for:
 - ☐ the project schedule;
 - ☐ the contractual, technical, and organizational aspects of the project; and
 - ☐ the assignment of responsibilities to the project groups.
2. Used to coordinate activities between the different engineering groups.
3. Readily available to all project staff and managers.
4. Updated to incorporate all intergroup commitments and changes to these commitments.
5. Regularly updated as the work progresses to reflect progress and plan changes at the project level, particularly when major project milestones are completed and when plans change significantly.
6. Reviewed and agreed to by all project engineering groups and the project manager.

Activity 4

Critical dependencies are identified and negotiated according to a documented procedure.

This procedure requires that:

1. Each critical dependency is explicitly defined, including:
 - ☐ the item to be provided,
 - ☐ who will provide it,
 - ☐ when it will be provided, and
 - ☐ the criteria for acceptance.
2. Critical dependencies are negotiated:
 - ☐ within the software engineering group (i.e., between subgroups),
 - ☐ between the software engineering group and other groups in the project and organization, and
 - ☐ between the software engineering group and the customer and end users.
3. Need dates and availability dates of critical dependency items are tied to the project schedule and the software schedule.
4. The agreement for each critical dependency is reviewed and approved by both the receiving group and the group responsible for providing the critical dependency item.
5. Critical dependencies are tracked on a regular basis and corrective actions are taken when appropriate.
 - ☐ Status and actual or projected completion are compared to the plan used to coordinate intergroup commitments.
 - ☐ Effects of late and early completions are evaluated for impacts on future activities and milestones.
 - ☐ Actual and potential problems are reported to the appropriate managers.

Activity 5 **Products produced as input to other project groups are reviewed by representatives of the receiving groups to ensure that the products match their needs.**

Activity 6 **All intergroup issues affecting software are documented, negotiated, and, if unresolved, reported to the appropriate managers.**

Examples of intergroup issues include:

- incompatible schedules,
- inadequate funding,
- technical risks,
- system-level design and requirements defects, and
- system-level problems.

Activity 7 **Periodic technical reviews and interchanges are held with the task leaders of the project groups.**

In these meetings, the task leaders:

1. Provide appropriate visibility of the product end users' needs and desires.
2. Monitor the technical activities of the project.
3. Ensure that the groups' interpretation and implementation of the technical requirements conform to the project requirements.
4. Ensure that commitments are being met and will continue to be met.
5. Ensure that technical risks and other technical issues are resolved in a timely manner.

Monitoring implementation

Monitor 1

Measurements are made and used to quantify and evaluate the status of the cost and schedule of the activities for intergroup coordination.

The following are tracked and compared against the initial estimate and budget:

1. Actual effort and other resources expended by the software engineering group for support to other project groups.
2. Actual effort and other resources expended by the nonsoftware engineering groups for support to the software engineering group.
3. Actual completion of specific tasks and milestones by the software engineering group to support the activities of other project groups.
4. Actual completion of specific tasks and milestones by the nonsoftware engineering group to support the activities of the software engineering group.

Verifying implementation

Verification 1

The activities for intergroup coordination are reviewed with senior management on a regular basis.

Refer to the software project tracking and oversight key process area for practices covering the oversight reviews with senior management.

Verification 2 **The activities for intergroup coordination are reviewed with the project manager on a regular basis.**

Refer to the software project tracking and oversight key process area for practices covering the status/coordination reviews with the project manager.

Verification 3 **The software quality assurance group audits the activities and products for intergroup coordination, and reports results as appropriate.**

At a minimum, the reviews and audits cover the process for reviewing and accepting products between project groups.

Refer to the software quality assurance key process area for practices covering the quality assurance reviews and audits.

Peer Reviews

a key process area for Level 3: Defined

Peer reviews involve a methodical examination of work products by the producer's peers to identify defects and areas where changes and improvements are needed. The specific work products that will undergo peer review are identified as part of the software planning activities. Peer reviews are performed following defined procedures. These procedures cover preparing for the review, conducting the review, reporting the results of the review, and certifying review readiness/completion criteria. Actions identified in the review findings are documented and tracked until they are resolved.

Goals

Goal 1	Product defects are identified and fixed early in the life cycle.
Goal 2	Appropriate product improvements are identified and implemented early in the life cycle.
Goal 3	The staff members become more effective through a better understanding of their work products and knowledge of errors that can be prevented.
Goal 4	A rigorous group process for reviewing and evaluating product quality is established and used.

Commitment to perform

Commitment 1 **The organization follows a written policy for implementing peer reviews.**

This policy requires that:

1. The organization identifies a standard set of work products that must be reviewed by peers.
2. Each project identifies the work products that will undergo peer review.

The list of products includes:

- operational software and support software,
 - deliverable and nondeliverable products,
 - software and nonsoftware products (e.g., documents), and
 - process specifications.

3. Peer reviews are guided or led by trained peer review leaders.
4. Peer reviews focus on the product being reviewed and not on the producer.
5. Results of the peer reviews are not used to evaluate the performance of individuals.

Ability to perform

Ability 1

Leaders of peer reviews receive required training.

Training is provided in:

1. The objectives, principles, and methods of peer reviews.
2. Planning and organizing a peer review.
3. Evaluating readiness and completion criteria for peer reviews.
4. Conducting and facilitating a peer review.
5. Reporting the results of the peer review.
6. Tracking and confirming rework to address the items identified in the peer review.

Refer to the training program key process area for practices covering training.

Ability 2

Reviewers who participate in peer reviews are trained in the objectives, principles, and methods of peer reviews.

Reviewers are trained in formal training courses, by trained peer review leaders, or by participating in reviews under the guidance of competent, experienced reviewers, as appropriate.

Ability 3 **Appropriate resources and funding for performing peer reviews are allocated for each product to be reviewed.**

Resources and funding are allocated for the efforts to:

1. Prepare and distribute the review materials.
2. Lead the review.
3. Review the materials.
4. Participate in the review meeting.
5. Rework the product based on the items identified in the review.

Activities performed

Activity 1 **Peer reviews are planned and the plans are documented.**

These plans:

1. Identify the work products that will undergo peer review.
 - ☐ The work products selected include the set specified in the organization's standard software process.
2. Specify the schedule of peer reviews.
3. Identify the trained peer review leaders for each review.
4. Identify the reviewers for each work product.
5. Are reviewed by the plan developers' peers.

(Activity 1)

The practices in this key process area govern these peer reviews.

Activity 2

Peer reviews are performed according to a documented procedure.

This procedure requires that:

1. Peer reviews are planned and led by trained peer review leaders.
2. Review materials are distributed to the reviewers in advance so they can adequately prepare for the reviews.
3. Readiness and completion criteria for the peer reviews are specified and enforced.
 - ☐ Issues in satisfying these criteria are reported to the appropriate managers.
4. Checklists exist and are used to evaluate the work products in a consistent manner.
 - ☐ The checklists are tailored to the specific type of product and review, and they address criteria such as compliance with standards and procedures, completeness, correctness, style, rules of construction, and maintainability.
 - ☐ The checklists are reviewed by the checklist developers' peers and potential users.

The practices in this key process area govern these peer reviews.

5. Actions identified in the peer reviews are tracked until they are resolved.

- (Activity 2)** 6. The successful completion of peer reviews, including the rework to address the items identified in the reviews, is used as a completion criterion for the associated activity.

Activity 3 Information on the conduct and results of peer reviews is recorded in an organizational database.

Information recorded includes:

1. Identification of the product reviewed.
2. Size of the product.
3. Amount of preparation time for each reviewer.
4. Length of time of the review.
5. Types and number of defects found and fixed.
6. Action items assigned and closed.
7. Effort expended to close the action items.

Monitoring implementation

Monitor 1 Measurements are made and used to quantify and evaluate the status of the peer review process.

The following items are measured and used to determine if the remaining peer reviews need to be replanned:

1. Preparation lead time.

- (Monitor 1)**
2. Preparation time per reviewer.
 3. Product size.
 4. Size and structure of the review team.
 5. Experience of the review team.
 6. Length of review meetings.
 7. Number of action items.
 8. Rework effort.

Verifying implementation

- Verification 1** **The software quality assurance group reviews and audits the activities and products for peer review, and reports results as appropriate.**
- At a minimum, the reviews and audits confirm that:
1. The planned peer reviews are conducted.
 2. The peer review leaders are adequately trained for their role.
 3. The reviewers are properly trained or experienced for their role.
 4. The processes for preparing for the peer reviews, conducting the peer reviews, and performing the follow-up actions are followed.
 5. Reporting of peer review data is complete, accurate, and timely.

(Verification 1)

Refer to the software quality assurance key process area for practices covering the quality assurance reviews and audits.

Process Measurement and Analysis

a key process area for Level 4: Managed

Process measurement and analysis involves taking measurements of the performance of the organization's standard software process (as instantiated by the projects), analyzing these measurements, and making adjustments to stabilize the process performance within acceptable limits. The process and the associated measurements are established as a baseline and are used to plan and control the process in quantitative terms.

Goals

Goal 1	The organization's standard software process is stable and under statistical process control.
Goal 2	The relationship between product quality, productivity, and product development cycle time is understood in quantitative terms.
Goal 3	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.

Commitment to perform

Commitment 1 **The organization follows a written policy for measuring and stabilizing its standard software process.**

This policy requires that:

1. The organization and projects implement a documented plan to bring the process under statistical control.
2. Sensitive data relating to individuals' performance are protected and access to these data is appropriately controlled.

Ability to perform

Ability 1 **There is an organizational team funded and staffed that focuses on process measurement and analysis.**

1. This team is either part of the group responsible for coordinating the organization's software process activities (e.g., software engineering process group) or its activities are closely coordinated with that group.
2. The software quality assurance group, software managers, and software task leaders are involved in the measurement activities.
3. Appropriate tools to collect, store, validate, analyze, and report measurement data are made available to the staff measuring and analyzing the process.

(Ability 1)

Examples of appropriate tools include:

- software source code analyzers,
- database systems,
- statistical analysis packages, and
- problem tracking packages.

Ability 2

Where feasible and practical, the software engineering environment is instrumented to record specific process and product data.

The product data referred to in these practices are product measurements used in analyzing the software process.

Ability 3

The staff members implementing or supporting process measurement and analysis receive required training to perform their measurement and analysis activities.

Training is provided to the staff members, according to their defined roles, in:

1. Modeling and analyzing the software process.
2. Selecting, collecting, and validating process measurement data.
3. Applying basic statistical methods and analysis techniques (e.g., estimation models, Pareto diagrams, and control charts).

Refer to the training program key process area for practices covering training.

Ability 4

The staff and managers of the groups involved with the software process receive orientation on the goals and value of the process measurement and analysis program.

Activities performed

Activity 1

A documented and approved plan is used as the basis for the organization's and projects' activities for process measurement and analysis.

Specifically, this plan:

1. Specifies the goals and objectives of the measurement program.
 - ☐ The goals and objectives of the measurement program are tied to the organization's strategic goals for product quality, productivity, and product development cycle time.
2. Describes the activities to be performed and the schedule for these activities.
 - ☐ In addition to current organizational needs, process measurement and analysis needs that are specific to the project and that may be useful to future efforts are included in this plan.
3. Identifies the groups and individuals responsible for the activities.
4. Specifies the resources required, including staff and tools.
5. Identifies the procedures to be followed.
6. Is maintained under configuration management.

(Activity 1)

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Activity 2

The organization's standard software process is the basis for selecting the data to be collected and the analyses to be performed.

The key attributes of this software process that are considered include:

1. The key process activities and their cause-effect relationships.
2. The major products and their relationships to each other and to the process.
3. The process control points and data collection points.

Activity 3

Process and product metrics are identified based on their usefulness to the organization and the projects.

1. The metrics support the overall goals and objectives of the measurement program.
2. The metrics and the data collection are consistent across the projects.
3. The metrics are chosen from the entire software life cycle (i.e., both the development and post-development stages).
4. The metrics cover the properties of the key process activities and major products.

(Activity 3)

5. The specific data items to be collected, their precise definitions, the intended use of each data item, and the life-cycle control points at which they will be collected are defined.

Examples of collected data items include:

- estimated/planned versus actual data on software size, cost, and schedule;
- quality measurements as defined in the software quality plan;
- peer review coverage and efficiency;
- test coverage and efficiency;
- number and severity of defects found in the software requirements;
- number and severity of defects found in the software code; and
- number and rate of closure on action items.

6. The metrics address specific issues or unknown properties of the process.

In some cases, metrics may be research oriented and should be explicitly designated as such.

7. The metrics are selected to support predefined analysis activities.

Activity 4

Process and product data are collected according to a documented procedure.

This procedure requires that the collected data items:

1. Are a natural result of the software activities where possible.
2. Are stored in the organization's software process database.

(Activity 4)

Refer to the organization process definition key process area for practices covering the setup and control of the software process database.

3. Are reviewed to ensure consistency and accuracy.
4. Are validated by independent means where possible.

Activity 5

The analysis of the selected process data is performed according to a documented procedure.

This procedure requires that:

1. The selected process measurements appropriately characterize the process they represent.
2. The expected values for mean and variance are specified for each process metric.
3. The measured values of each process metric are compared to the expected values of the mean and variance.
4. The specific data analysis activities are performed according to a documented description of the analysis.
 - ☐ The description of the analysis activities covers the input data required, tools used, data manipulations performed, information to be derived, and decision criteria used.

(Activity 5)

5. Based on the results of the analyses, changes are made, where appropriate, to the following elements to bring them within defined acceptable limits:
 - ☐ the software process specifications,
 - ☐ the expected values of mean and variance,
 - ☐ the data collection requirements, and
 - ☐ the analysis performed.

Activity 6

The results from the process data analyses are used to bring the organization's standard software process and its critical subprocesses under statistical process control.

1. The overall performance of the organization's standard software process is controlled and stabilized.
 - ☐ Measurements of the key products and process activities across the standard software process are identified, collected, and analyzed.
 - ☐ Appropriate adjustments are made to bring the actual process performance in line with the expected performance.

Refer to the organization process definition key process area for practices covering the organization's standard software process.

2. When the organization's standard software process is stabilized, it and the process measurements are baselined.
 - ☐ The process metrics and actual measurements for the process are baselined.
 - ☐ Upper and lower limits for the process metrics are defined and documented.
3. When the organization undertakes a project or effort that is substantially different from past efforts, the expected process measurements from the new project are used to recalibrate and re-establish the process performance baseline.

(Activity 6)

Examples of substantially different efforts include:

- new application domains,
- use of radically different technologies, and
- significant increase in the size of the application.

Activity 7

The process performance baseline for the organization's standard software process is monitored on a regular basis and updated as appropriate.

1. Change activity for the standard software process is tracked and analyzed over time to determine if the process is approaching a stable state.
2. Process defects are identified and fixed.
3. Process performance trends are examined to predict likely problems or opportunities for improvements.

Examples of areas that are likely sources of problems include:

- estimating and planning items,
- activities performed early in the life cycle such as requirements analysis,
- major documentation items,
- items and activities that have been prone to error in the past,
- activities for implementing changes and fixing problems, and
- labor-intensive activities.

(Activity 7)

Examples of areas that are likely opportunities for improvement include:

- activities that other projects and organizations have successfully automated,
- nondeliverable and support items and activities such as training and tools,
- quality oriented activities such as peer reviews and testing, and
- labor intensive activities.

Activity 8

The results of the measurement and analysis activities are monitored on a regular basis and appropriate adjustments are made to keep the process performance baseline in line with the expected performance.

The process performance baseline is adjusted to reflect changes made to the process specification and to reflect a better understanding of the process.

Activity 9

Process analysis reports are prepared and distributed to the appropriate groups.

1. The results of the data analysis are reviewed with those affected by the data before they are reported to anyone else.
2. Software management, software task leaders, and senior management receive regular reports, appropriate for their needs.
3. The software quality assurance group receives regular reports to help it determine which software activities and products need to be reviewed.
4. The project manager, senior managers, software managers, and software task leaders receive specialized reports on request.

Monitoring implementation

Monitor 1

Measurements are made and used to determine the cost and schedule status of the activities for process measurement and analysis.

The cost over time and the accomplishment of schedule milestones for process measurement and analysis activities are compared to the approved plan; status and deviations that require management attention are reported to the appropriate managers.

Verifying implementation

Verification 1

The results of the data analysis are reviewed with the software engineering staff, software managers, and users of the analysis reports to ensure that the results are valid and useful.

Verification 2

The group responsible for coordinating the organization's software process activities (e.g., software engineering process group) regularly reviews the activities for process measurement and analysis and initiates appropriate corrective actions.

Specifically, corrective actions are taken if:

1. The needed data do not exist.
2. The needed data are not collected.
3. The data collected are not needed.
4. The data collected do not support the goals and objectives of the measurement program.

- (Verification 2)**
- 5. The cost of collecting the data is not justified by the usefulness of the data.
 - 6. The data are collected at the wrong point in the life cycle.
 - 7. The data are inaccurate or incorrect.
 - 8. The data are not timely.
 - 9. The confidentiality of the data is not properly protected.

Verification 3 **The organization's and projects' activities for process measurement and analysis are reviewed with senior management on a regular basis.**

Refer to the software project tracking and oversight key process area for practices covering the oversight reviews with senior management.

Verification 4 **The project's activities for process measurement and analysis are reviewed with the project manager on a regular basis.**

Refer to the software project tracking and oversight key process area for practices covering the status/coordination reviews with the project manager.

Verification 5 **The software quality assurance group reviews and audits the activities and products for process measurement and analysis, and reports results as appropriate.**

(Verification 5) At a minimum, the reviews and audits determine whether:

1. The plans and schedules for the process measurement and analysis activities are followed.
2. The procedures for process measurement and analysis are followed.

Refer to the software quality assurance key process area for practices covering the quality assurance reviews and audits.

Quality Management

a key process area for Level 4: Managed

Quality management involves defining software quality goals, establishing plans to achieve these goals, and monitoring and adjusting the software plans, activities, and quality goals to improve customer and end-user satisfaction. Quantitative product quality goals are established based on the needs of the organization, customer, and end users. Plans and process quality goals are established and the project's software process is specifically adjusted to achieve the product quality goals. The software activities and results are assessed against the quality objectives on a regular basis, and corrective actions are taken to bring forecasted process and product quality in line with the goals.

Goals

Goal 1	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.
Goal 2	Measurable goals for process quality are established for all groups involved in the software process.
Goal 3	The software plans, design, and process are adjusted to bring forecasted process and product quality in line with the goals.
Goal 4	Process measurements are used to manage the software project quantitatively.

Commitment to perform

Commitment 1 The organization follows a written policy for managing quality on software projects.

This policy requires that:

1. The organization defines and documents its commitment to improve product quality continuously.

Improvements to the process that increase product quality are a top priority of the organization.

Each new product release should be measurably better than its predecessor or leading competitor.

2. The projects define and quantitatively manage the measures from their defined software processes.
3. The projects define and monitor their product quality goals.
4. The responsibilities for quality are defined for each group involved in the software process and criteria are established to enable the group to check success in achieving quality.
5. Management takes action to prevent defects from recurring whenever possible.

Ability to perform

Ability 1

Appropriate resources and funding for quality management are provided.

1. Specialty engineers in areas such as safety and reliability are available to assist in setting the quality goals and reviewing progress against the goals.
2. Appropriate tools for measuring, tracking, and analyzing quality are made available to the staff responsible for the quality management activities.

Examples of appropriate tools include:

- data collection tools,
- database systems,
- spreadsheet programs,
- life-cycle simulators,
- statistical analysis tools, and
- code audit tools.

Ability 2

The staff implementing and supporting quality management receives required training to perform the quality management activities.

Training is provided, according to staff members' roles, in:

1. Planning quality commitments and goals for the product.
2. Measuring product and process quality.
3. Controlling product quality using the defined software process.

(Ability 2)

Refer to the training program key process area for practices covering training.

Ability 3

The staff and managers of the groups involved in the software process receive required training on software quality management.

Training is provided in:

1. The goals and benefits of quantitatively managing quality.
2. The metrics for software process and product quality.
3. Planning and controlling software product quality.

Refer to the training program key process area for practices covering training.

Activities performed

Activity 1

The project develops strategies to satisfy the quality needs of the organization, customer, and end users.

1. The project focuses on understanding the quality needs of the organization, customer, and end users as appropriate.

The end users referred to in these practices are the customer-designated end users or representatives of the end users.

(Activity 1)

Examples of ways to measure the customer's and end-users' quality needs include:

- surveys,
- focus groups, and
- product evaluations by users.

2. The quality needs and priorities of the organization, customer, and end user are traceable to functional requirements, quality goals, and software processes.

An example of a method to trace these needs and priorities is Quality Function Deployment (QFD).

Refer to "The House of Quality" by J.R. Hauser and D. Clausing (*Harvard Business Review*, May-June 1988, pp. 63-73) for additional information on Quality Function Deployment.

3. The ability of the software process to satisfy the quality goals is assessed and documented.

Activity 2

A documented and approved software quality plan for the project is the basis for the project's activities for software quality management.

The plan:

1. Supports the quality programs of the organization and the company.
2. Is based on plans for previous or current projects in the organization, where appropriate.

(Activity 2)

3. Is coordinated with senior management.
4. Identifies the points in the process where the process quality and product quality are checked.
 - ☐ Software product quality goals are checked at each process stage in the life cycle.
 - ☐ Software process quality goals are checked for each task in the project's defined software process.
5. Describes how the project will improve on past quality performance.
6. Specifies the software activities used to check product quality.

Examples of software activities to check product quality include:

- peer reviews,
- prototype development,
- product simulation, and
- testing.

7. Specifies product quality goals for intermediate products (e.g., designs), as appropriate, and for final products.
8. Defines the actions to be taken when the product's quality is projected not to meet the quality goals.
9. Is reviewed by the plan developers' peers.

Refer to the peer review key process area for practices covering peer reviews.

10. Is reviewed and agreed to by all affected groups.

(Activity 2)

Examples of affected groups include:

- software engineering,
- software testing,
- software quality assurance,
- software configuration management , and
- the group responsible for coordinating the organization's software process activities (e.g., software engineering process group).

11. Is reviewed and agreed to by the customer and end users, or their representatives.
12. Is maintained under configuration management.

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

Activity 3

Quantitative product quality goals are defined and revised throughout the software life cycle.

1. Attributes of product quality that describe how well the product will perform or how well it can be developed, maintained, and enhanced are identified.

(Activity 3)

An example of such attributes of product quality include nonfunctional requirements such as:

- reliability,
- maintainability,
- availability, and
- usability.

2. Activities are conducted to identify product quality metrics used to quantify these attributes of product quality.

Examples of activities to identify product quality metrics include:

- reviewing prior performance data and customer requirements,
- developing prototypes,
- expressing intermediate software products in formal representations,
- using formal software engineering methods, and
- conducting tests.

3. For each attribute of product quality, measurable, numeric values based on the required and desired values are selected as product quality goals.

Examples of product quality goals for reliability are:

- the mean-time-between-failure as specified in the requirements,
- the mean-time-between-failure that must be achieved (as determined by analysis and experimentation), and
- the mean-time-between-failure that is planned to be achieved.

(Activity 3)

4. Product quality goals are documented in the software quality plan and include:
 - ☐ the critical attributes that if not met would make the product undesirable or not needed,
 - ☐ the attributes required by the contract with the customer, and
 - ☐ the attributes which are planned to be met.
5. Product quality goals are revised as understanding of the product and understanding of the organization's, customer's, and end-users' needs evolve.

Activity 4

Quantitative process quality goals are established for the software project.

1. The project's defined software process is designed to address the product's quality goals specifically.
2. The process metrics and their desired values are identified.

Examples of process metrics include:

- staff hours per defect,
- average defects per page,
- average defects per page per process stage, and
- average defects per number of pages reviewed.

An example of a desired value is the number of document pages of design that can be effectively reviewed in a peer review.

3. The desired values are documented as process quality goals in the software quality plan for the project.

- (Activity 4) 4. Each group involved in the software process reviews and agrees to the process metrics and desired values to which that group contributes.

Activity 5 Software product quality goals flow down to subcontractors.

Refer to the software subcontract management key process area for practices covering managing a subcontract.

Activity 6 Quantitative quality goals for software requirements are established and tracked.

1. Software requirements are analyzed for quality.

Examples of quality attributes for software requirements are:

- consistency,
- feasibility, and
- testability.

Examples of methods of analyzing software requirements include:

- peer reviews,
- prototype development,
- simulation, and
- traceability analysis.

2. Appropriate actions, consistent with the software quality plan, are taken to bring the quality measurements of the software requirements in line with the goals.

Activity 7

Quantitative quality goals for software design are established and tracked.

1. The software design is analyzed for quality.

Examples of quality attributes for design are:

- complexity,
- maintainability,
- completeness, and
- consistency.

Examples of formal methods for analyzing the quality of the software design include peer reviews and proof-of-correctness techniques such as:

- invariant assertions,
- weakest preconditions, and
- functional correctness.

2. Appropriate actions, consistent with the software quality plan, are taken to bring the quality measures of the software design in line with the goals.

Activity 8

Alternative software designs are considered to meet software product quality goals and software requirements.

1. Analyses are performed to determine how much a design contributes positively or negatively to product quality goals.
2. Techniques are used to optimize the design for the given requirements and quality goals.

(Activity 8)

An example of a quantitative method for optimizing software design is Taguchi's method for robust design.

Refer to "Robust Quality," by G. Taguchi and D. Clausing (*Harvard Business Review*, January-February 1990, pp. 65-75) for additional information on Taguchi's method for robust design.

Activity 9

Quantitative quality goals for software code are established and tracked.

Examples of quality attributes for code are:

- complexity,
- readability,
- portability, and
- execution speed.

Activity 10

Quantitative quality goals for formal software tests are established and tracked.

Activity 11

When quality goals are discovered to conflict (one goal cannot be achieved without compromising another goal), the software requirements, software design, software development plan, and software quality plan are revised to reflect the necessary tradeoffs.

1. The cost for achieving the quality goals is analyzed as part of software project planning.
2. The cost for achieving the quality goals for a given design is analyzed.

(Activity 11)

3. Alternatives are considered in light of long-term business strategy as well as short-term priorities.
4. The customer and end users, or their representatives, participate in quality tradeoff decisions.

Activity 12

The groups involved in the software process review, agree to, and work to meet the project's quality goals for its process and products.

At the beginning of a software task, the staff performing the task:

1. Reviews the process and product quality goals.
2. Determines the quality goals applicable to the task.
3. Identifies its plans to achieve the quality goals.
4. Reviews changes made to the process to meet the quality goals.

An example of a change is revising a peer review checklist to address defects found to escape peer reviews.

Activity 13

Process data are monitored to identify actions needed to satisfy the process quality goals.

1. Process measurements are analyzed against their planned goals and control limits.

An example of establishing control limits is to calculate the historical deviation from the mean performance of the process.

(Activity 13)

The comparisons of the measured values with the planned goals and upper and lower control limits are used in decisions involving tradeoffs among product schedule, quality, and functionality.

2. Process data are recorded in an organizational process database.

Refer to the organization process definition key process area for practices covering the organization's software process database.

Activity 14

The quality of the project's products are compared against the product's quality goals on a regular basis.

Activity 15

Corrective actions are taken by the groups involved in the software process when the quality measurements indicate process or product problems.

1. Factors causing the problems are identified, where possible, and are corrected or eliminated.

Examples of possible actions include:

- executing contingency plans,
- following alternatives processes, or
- changing the project's defined software process.

2. The status and results of corrective actions are tracked and periodically reviewed with the project manager and project software manager.

Monitoring implementation

Monitor 1

Measurements are made and used to quantify and evaluate the process for quality management.

1. The cost of poor quality is calculated, based on the known quality measurements to whatever degree of accuracy they can be collected.
2. The costs for achieving the quality goals are calculated.
3. High-leverage process and product quality goals are identified for tracking.

Verifying implementation

Verification 1

Senior management reviews process quality goals against the organization's quality policies.

1. Software quality plans are reviewed by senior management.
2. Software quality plans are updated at the start of the project at major project milestones, and whenever the software requirements change significantly.

Verification 2

The activities for managing quality are reviewed with senior management on a regular basis.

The senior management oversight reviews are specified in the software project tracking and oversight key process area.

Verification 3 **The activities for managing quality are reviewed with the project manager on a regular basis.**

The project manager status/coordination reviews are specified in the software project tracking and oversight key process area.

Verification 4 **The software quality assurance group reviews and audits the activities and products for managing quality, and reports results as appropriate.**

At a minimum, the reviews and audits cover:

1. The preparation of the software quality plan.
2. The process for establishing and tracking the quality goals.
3. The process followed in implementing corrective actions when the performance of the project's defined software process is outside its control limits.

The software quality assurance reviews and audits are specified in the software quality assurance key process area.

Defect Prevention

a key process area for Level 5: Optimizing

Defect prevention involves analyzing defects that were encountered in the past and taking action to prevent the injection of these types of defects in current and future project activities. Software activities are systematically reviewed by those who perform them to identify the defects that were encountered, to understand the root causes of the defects, and to determine the implications of the defects on future activities. Trends are analyzed to determine the kinds of defects that were encountered in the past. Defects which are likely to recur are identified, and specific actions are taken to prevent them.

Goals

Goal 1 Sources of product defects that are inherent or repeatedly occur in the software process activities are identified and eliminated.

Commitment to perform

Commitment 1 The organization follows a written policy governing defect prevention activities. This policy requires that defect prevention activities are:

1. Implemented across the organization to improve the software process.
2. Included in each project's software development plan.

Commitment 2 Management supports and participates in defect prevention activities.

Specifically, management:

1. Establishes long-term plans and commitments for funding, staffing, and other resources for defect prevention.
2. Allocates the resources needed for the defect prevention activities.
3. Handles management actions identified as a result of the defect prevention activities.
4. Reviews the results of the defect prevention activities to ensure the effectiveness of these activities and resolve management issues.

Ability to perform

Ability 1 The organization has a team which coordinates defect prevention activities.

This team is either part of the group responsible for coordinating the organization's software process activities (e.g., software engineering process group) or its activities are closely coordinated with that group.

Refer to the organization process focus key process area for practices covering the responsibilities for coordinating the organization's software process activities.

Ability 2 **Each project has a team funded and staffed to coordinate defect prevention activities for the project.**

This team is closely tied to the team or individuals responsible for defining and maintaining the project's defined software process.

Refer to the integrated software management key process area for practices covering the responsibilities for the project's defined software process.

Ability 3 **Appropriate resources and funding are provided for defect prevention activities.**

1. Defect prevention activities, as appropriate, are planned into each person's responsibilities.

Defect prevention activities include:

- task kickoff meetings,
- causal analysis meetings,
- reviewing and planning proposed actions, and
- implementing actions.

2. Management participation in the defect prevention activities is planned.
3. Each project is represented on the team coordinating defect prevention activities for the organization.
4. Members of the teams coordinating defect prevention activities possess, or have ready access to, skills and expertise enabling them to perform the defect prevention investigations and actions.

(Ability 3)

Members of the the teams coordinating defect prevention activities are usually assigned to this team on a part-time basis and have other project software engineering activities as their primary responsibility.

Examples of skills and expertise include:

- software process definition,
- training,
- tools and methods,
- management of involved products,
- software quality assurance,
- software configuration management, and
- system test.

5. Appropriate tools are made available to the action teams.

Examples of tools include statistical analysis tools and database systems.

Ability 4

The software engineering staff and managers receive required training to perform their defect prevention activities.

Training is provided, according to a person's defined role, in:

1. Defect prevention methods.
2. Conduct of causal analysis meetings.
3. Statistical methods.

(Ability 4)

Examples of statistical methods include cause/effect diagrams and Pareto analysis.

Refer to the training program key process area for practices covering training.

Ability 5

Defect prevention meetings and other defect prevention activities are incorporated in the project's software development plan and schedules.

Activities performed

Activity 1

At the beginning of a software task, the members of the team performing the task meet to prepare for the activities of that task and the related defect prevention activities.

These kickoff meetings cover:

1. The software process, standards, procedures, methods, and tools applicable to the task, with emphasis on recent changes.

Changes may be implemented as an experiment to evaluate a recommendation from a previous causal analysis meeting.

2. The inputs required and available for the task.
3. The outputs to be produced with examples, if available.
4. The methods to be used to evaluate and validate the outputs.

(Activity 1)

5. The methods to be used to validate adherence to the software process.
6. A list of errors that are commonly made or introduced during the current stage and recommended preventive actions for these errors.
7. The team assignments.
8. The task schedule.
9. The quality expectations and goals for the task and project.

Activity 2

Each team that performs a software task conducts a causal analysis meeting shortly after the task is completed; meetings are also conducted during the task if and when the number of defects uncovered warrants the additional meetings.

1. The meetings are led by a person trained in conducting causal analysis meetings.
2. Defects are identified and analyzed to determine their root causes.

An example of a method to determine root causes is cause/effect diagrams.

3. The defects are assigned to categories of root causes.

(Activity 2)

Examples of defect root cause categories include:

- inadequate training,
- breakdown of communications,
- not accounting for all details of the problem, and
- making mistakes in manual procedures (e.g., typing).

4. Proposed actions to prevent the future occurrence of identified defects and similar defects are developed and documented.

Examples of areas for proposed actions include modifications to:

- the process,
- training,
- tools,
- methods,
- communications, and
- products.

5. Trends that indicate broad problems are identified and documented.

Examples of trends include frequent errors made in invoking a certain system function or frequent errors made in a related group of software modules.

6. The results of the meeting are recorded in a database for use by the organization and other projects.

Activity 3 **Periodic causal analysis meetings are conducted after products are released to the customer.**

These meetings adhere to the practices specified for the causal analysis meetings held at the completion of a task.

Activity 4 **For software tasks of long duration, periodic in-process meetings, conforming to the practices for task kickoff meetings and causal analysis meetings, are conducted.**

Activity 5 **Each of the teams assigned to coordinate defect prevention activities meets regularly to review and coordinate implementation of action proposals from the causal analysis meetings.**

Specifically, the teams:

1. Review the output from the causal analysis meetings and select actions that will be addressed.
2. Review results of defect prevention experiments and take actions to incorporate the results of successful experiments into the rest of the project.

Examples of defect prevention experiments include using a temporarily modified process and using a new tool.

3. Review action items that have been assigned to them by other action teams in the organization and select actions that will be addressed.
4. Review actions taken by the other teams in the organization to assess whether these actions can be applied to their activities and processes and to take appropriate actions.

(Activity 5)

5. Identify action items which need to be addressed at a higher level in the organization or company and reassign these items.
6. Perform a preliminary analysis of the action items and set their priorities.

Priority is usually nonrigorous and is based on an understanding of:

- the causes of defects,
- the implications of not addressing the defects,
- the cost to implement process improvements to prevent the defects, and
- the expected impact on quality.

An example of a technique used to set priorities for the improvements is Pareto analysis.

7. Document their rationale for decisions; rationale for rejections are provided to the submitters of the action proposal.
8. Assign responsibility for implementing the action items.
 - ☐ Implementation of the action items includes making immediate changes to the activities, etc., that are within the purview of the team and arranging for other changes.
 - ☐ Members of the team usually implement the action items, but, in some cases, the team can arrange for someone else to implement an action item.
 - ☐ Management actions are implemented by the manager on the team.
9. Document process improvement proposals for the organization's standard software process and the projects' defined software processes as appropriate.

(Activity 5)

Refer to the process change management key process area for practices covering process improvement proposals.

The submitters of the action proposal are designated as the submitters of the process improvement proposals.

10. Track the status of the action items.
11. Review and verify completed action items before they are closed.
12. Ensure that significant efforts and successes in preventing defects are recognized.

Activity 6

Revisions to the organization's standard software process and to the projects' defined software processes resulting from defect prevention actions are incorporated according to a documented procedure.

Refer to the organization process definition key process area for practices for revising the organization's standard software process.

Refer to the integrated software management key process area for practices for revising a project's defined software process.

Activity 7

A database for documenting, tracking, and coordinating defect prevention actions across the organization is used according to a documented procedure.

This procedure requires that the database be used to:

1. Document proposed actions identified in causal analysis meetings.

Examples of data that are in the description of a proposed action item include:

- action item originator,
- description of the defect,
- description of the defect cause,
- defect cause category,
- stage the defect was injected,
- stage the defect was identified,
- description of the action item, and
- action item category.

2. Document the plan and status for each action item, as appropriate.

Examples of information that is in the plan for an action item include:

- the person responsible for implementing it,
- description of the areas affected by it,
- individuals who are to be kept informed of its status,
- the next date its status will be reviewed, and
- rationale for key decisions.

3. Document the results of implemented action items.

(Activity 7)

Examples of information that is documented for the implementation results for an action item include:

- a description of implementation actions,
- the time and cost for identifying the defect and correcting it, and
- the estimated cost of not fixing the defect.

4. Document the reassignment of project action items, as appropriate, to the team coordinating defect prevention activities for the organization.
5. Document the reassignment of organizational action items, as appropriate, to a higher level team.
6. Ensure the integrity of the defect prevention data.
 - ☐ The software engineering staff and managers can view the contents of the database.
 - ☐ Updating of action items and specific fields of the action items is restricted to individuals according to their responsibilities.

Activity 8

The project's software engineering staff and managers receive feedback on the status and results of the organization's and project's defect prevention activities on a regular basis.

The feedback provides:

1. A summary of the major defect categories.
2. The frequency distribution of defects in the major defect categories.
3. Significant innovations and actions taken to address the major defect categories.

(Activity 8)

4. A summary status of the proposed, open, and completed action items.

Examples of means to provide this feedback include:

- electronic bulletin boards,
- newsletters, and
- information flow meetings.

Refer to "Experiences with Defect Prevention" by R. G. Mays, C. L. Jones, G. J. Holloway, and D. P. Studinski (*IBM Systems Journal*, Vol. 29, No. 1, 1990, pp. 4-32) for additional information on defect prevention.

Monitoring implementation

Monitor 1

Measurements are made and used to determine the cost and schedule status of the defect prevention activities.

1. The costs of defect prevention activities (e.g., causal analysis meetings and implementing action items) are tracked over time.
2. The time and cost for identifying the defects and correcting them are compared to the estimated cost of not correcting the defects.
3. Profiles are maintained measuring the number of action items proposed, open, and completed.

Monitor 2 **Measurements are made and used to determine the quality of the defect prevention process.**

The number of defects injected in each stage is measured and compared over time and over releases of similar products.

Verifying implementation

Verification 1 **The defect prevention activities are reviewed with the project managers and the senior managers on a regular basis.**

These reviews cover:

1. A summary of the major defect categories and the frequency distribution of defects in these categories.
2. A summary of the major action categories and the frequency distribution of actions in these categories.
3. Significant actions taken to address the major defect categories.
4. A summary status of the proposed, open, and completed action items.
5. A summary of the effectiveness of and savings attributable to the defect prevention activities.
6. The actual cost of completed defect prevention activities and the projected cost of planned defect prevention activities.

Verification 2

The software quality assurance group reviews and audits the activities and products for defect prevention, and reports results as appropriate.

At a minimum, the reviews and audits confirm that:

1. The software engineering staff and managers are trained for their defect prevention roles.
2. The task kickoff meetings and causal analysis meetings are properly conducted.
3. The process for reviewing and implementing proposed action items is followed.

Refer to the software quality assurance key process area for practices covering the quality assurance reviews and audits.

Technology Innovation

a key process area for Level 5: Optimizing

Technology innovation involves identifying, selecting, and evaluating new technologies, and incorporating the appropriate technologies into the organization's processes. A group acts as the focus for introducing technology innovations into the organization. By maintaining an awareness of software technology innovations throughout the world and systematically evaluating and experimenting with them, the organization selects appropriate technologies to improve its productivity and product quality. Pilot efforts are performed to assess new and unproven technologies before they are introduced across the organization. With appropriate sponsorship of the organization's management, the selected technologies are incorporated into the organization's process.

Goals

Goal 1	The organization has a software process and technology capability to allow it to develop or capitalize on the best available technologies in the industry.
Goal 2	Selection and transfer of new technology into the organization is orderly and thorough.
Goal 3	Technology innovations are tied to quality and productivity improvements of the organization's standard software process.

Commitment to perform

Commitment 1 The organization follows a written policy for improving its technology capability.

This policy requires that the organization:

1. Establishes technology innovation objectives in its strategic and operating plans.
2. Implements a documented plan to address the technology innovation objectives.

Commitment 2 Senior management sponsors the organization's technology innovation activities.

Specifically, senior management:

1. Defines a strategy for becoming an industry leader in software technology.
 - ☐ The strategy addresses the customers' and end-users' needs and desires, as appropriate, and the organization's goals for product quality, productivity, and product development cycle time.
2. Coordinates with the organization's managers in defining their departments' goals and approaches for accomplishing the organization's strategy.
3. Makes a commitment to the technology innovation effort that is visible to the organization's staff and managers.
4. Establishes long-term plans and commitments for funding, staffing, and other resources.

Commitment 3 Senior management oversees the organization's technology innovation activities.

Specifically, senior management:

1. Helps to establish technology innovation policies and reviews and approves these policies.
2. Allocates resources for technology innovation activities.
3. Helps relate organizational strategies and objectives to technology innovation strategies.
4. Participates in establishing the technology innovation plans.
 - ☐ Senior management coordinates technology innovation requirements and issues with higher-level corporate staff and managers as appropriate.
 - ☐ Senior management coordinates with the organization's managers to secure the managers' and staff's support and participation.

Ability to perform

Ability 1 The organization funds and staffs a group that focuses on technology innovation.

1. The group is either part of the group responsible for coordinating the organization's software process activities (e.g., software engineering process group) or its activities are closely coordinated with that group.

A common example of a group that focuses on technology innovation is a technology support group.

(Ability 1)

2. The group coordinates and provides assistance in:
 - ☐ exploring potential areas for applying new technology;
 - ☐ selecting and planning for new technologies;
 - ☐ acquiring, installing, and customizing new technologies;
 - ☐ communicating and coordinating with related research and development activities within the company; and
 - ☐ communicating with the technology suppliers on problems and enhancements.
3. Experienced staff members with expertise in specialized areas are available to this group to help in evaluating, planning, and supporting technology innovation initiatives.

Examples of specialized areas include:

- workstations,
- computer hardware,
- software reuse,
- computer-aided software engineering (CASE) technology,
- measurements,
- software engineering environments,
- formal methods, and
- programming languages.

Ability 2

The software engineering environment provides data collection and analysis support for evaluating technology innovations.

Specifically, the environment provides the capability to:

1. Automatically record selected process and product data.
2. Support data analysis.
3. Graphically display selected data.

Ability 3

Appropriate data on the software processes and products are available to support analyses performed to quantitatively evaluate and select technology innovations.

Examples of process and product data include:

- resource expenditures and productivity by project, process stage, tools and methods used, program category, degree of program modification, etc.;
- schedule time by project, process stage of each project, program category, program size, degree of program modification, etc.;
- peer review data, including defect data and review efficiencies;
- defect data showing stage introduced, stage removed, type, cause, severity, and time and effort to fix;
- change activity, including amount of code produced, amount of documentation produced, etc.;
- data on the defect fix activities, including the identification of the defects, the product version that the defect fix was implemented, and identification of defects introduced in implementing each defect fix; and
- density of defects by project, product type, specific product, and specific subproduct (e.g., program modules).

Ability 4

The staff members focusing on technology innovation receive required training to perform their activities.

Training is provided in:

1. The organization's standard software process.
2. Technology transfer and change management.
3. Software process improvement.
4. Tools and methods used by the organization.

(Ability 4)

5. Analytical and support facilities provided by the software engineering environment.
6. Principles of statistical quality management.

Refer to the training program key process area for practices covering training.

Activities performed

Activity 1

The organization develops and maintains a plan for technology innovation.

This plan:

1. Covers the assigned responsibilities and resources required, including staff and tools.
2. Defines the long term technical strategy for automating and improving the organization's software activities and enhancing the organization's market position.
3. Identifies the processes to be followed in performing the organization's technology innovation activities.
4. Describes the plans to introduce new technologies to address specific needs of the organization and projects.
 - ☐ Process areas that are potential areas for technology innovation are identified.
 - ☐ Approaches for identifying opportunities for technology innovation are identified.

(Activity 1)

- ☐ The specific planned or candidate tools and technologies are identified.
 - ☐ Where appropriate, the life-cycle plan for the planned tools and technologies is estimated, from introduction to replacement.
 - ☐ The make/buy tradeoff studies are documented.
 - ☐ Plans for assessing unproven candidate technologies are defined.
 - ☐ The acquisition and installation plans are defined.
 - ☐ The initial training, continuing training, and consultation support are planned.
5. Is reviewed by the plan developers' peers.

Refer to the peer review key process area for practices covering peer reviews.

6. Is reviewed and agreed to by the managers of the software projects before the effort is started.

Activity 2

The organization's and projects' staff and managers are kept aware of appropriate new technologies.

1. Available new technologies that may be appropriate to the organization's and projects' needs are identified.
 - ☐ A periodic search is made to identify commercially available tools that meet the organization's identified and anticipated needs.
 - ☐ Systematic efforts are made to maintain awareness of leading relevant technical work and trends of new technologies.
 - ☐ Systematic efforts are made to review the technologies used by other organizations within the company and by other companies in related businesses and to compare these technologies to those used within the organization.

(Activity 2)

- ☐ Areas where new tools, methods, or technologies have been used successfully are identified, and data and documentation of experience on using them are collected and reviewed.
- 2. Available new technologies are screened and evaluated to determine their applicability to the organization's and projects' current and future needs.
- 3. Information on new technologies which pass the initial screening are disseminated throughout the organization and projects as appropriate.
- 4. Information on advanced technologies in use in parts of the organization are disseminated throughout the organization as appropriate.

Activity 3

The group focusing on technology innovation (e.g., technology support group) is involved with the organization's staff and managers in identifying areas of technology innovation.

This group:

- 1. Participates in analyzing the software process activities to identify potential areas of technology innovation.
- 2. Solicits and encourages suggestions for technology innovation.
- 3. Tracks the actions taken and reports on the status of the technology suggestions.

Activity 4

The organization analyzes its standard software process to identify areas that need or could benefit from new technology.

- 1. Analyses are made of each process stage to determine areas where new tools, methods, or technologies would be most helpful to support process development and/or implementation.

(Activity 4)

2. The required technology innovations and the economics of those innovations are determined and documented.
3. The relationship of the tools and technology to the organization's standard software process is defined.
4. The expected outcomes of the technology innovations are qualitatively and quantitatively defined.
5. The need for piloting each potential technology innovation is determined.
6. The priority order of the candidate new technologies is determined.

Activity 5

A documented procedure is followed for selecting and acquiring technologies for the organization and projects.

This procedure requires that:

1. Requests for the acquisition of new tools and methods are documented and approved.
 - ☐ Management approval is required for activities with projected expenses above a predefined level.
2. Preliminary cost/benefit analyses are performed for the potential technology innovations.
3. Predefined and approved selection criteria are used to identify the highest potential benefits.
4. Requirements and plans for the selected technology innovations are defined, documented, and approved.
 - ☐ Where practical, the expected life span and plans for replacement/upgrade are estimated.

(Activity 5)

- ☐ Where appropriate, tradeoff studies are performed, reviewed, and documented to determine whether the technology should be developed by the company or procured from outside the company.
- ☐ Where appropriate, the plan provides for installing the technology innovation on a pilot basis to determine its effectiveness and economic benefits.
- ☐ The requirements and plans are approved by the managers of the affected groups and the group responsible for technology innovation (e.g., technology support group).

Activity 6

Pilot efforts for improving technology are conducted, where appropriate, before technology is introduced on a broad scale.

1. These pilot efforts are conducted to determine the feasibility and economics of untried or advanced technologies.
2. The plans for the pilot effort are defined, documented, reviewed by the plan developers' peers, and approved before the effort begins.
 - ☐ The plan covers the objectives, evaluation criteria, and activities for the pilot effort.
 - ☐ The plan for conducting the pilot effort is reviewed and approved by the managers of the affected groups.

Refer to the peer review key process area for practices covering peer reviews.

3. The group focusing on technology innovation (e.g., technology support group) provides consultation and assistance to the pilot effort.
4. The pilot effort is performed with respect to the process environment for which it is intended.

(Activity 6)

5. The results of the pilot effort are collected, analyzed, and documented.
 - ☐ Lessons learned and problems encountered during the effort are documented.
 - ☐ The benefits and impacts of broader use in the organization are estimated. The uncertainty in these estimates is assessed.
 - ☐ A decision is made whether to terminate the effort, proceed with broad-scale implementation of the technology, or replan and continue the pilot effort.
 - ☐ The plan for using the results of the pilot effort is defined and documented.

Activity 7

Appropriate new technologies are incorporated into the organization's standard software process and the projects' defined software processes according to a documented procedure.

This procedure requires that:

1. For significant technology innovations, senior management coordinates with the organization's managers to secure the managers' and staff's acceptance and support.
2. The resources needed to install, maintain, and support the technologies are established and funded.
3. The strategy for collecting data to measure and track the process performance changes due to the new technologies is documented, reviewed, and approved.
 - ☐ This strategy is agreed to by the staff and managers responsible for implementing the software processes affected by the new technologies.
4. Technology changes that affect the stability of the software process are maintained under configuration management.

(Activity 7)

Refer to the software configuration management key process area for practices covering configuration management.

Different levels of configuration management are implemented as appropriate.

5. The software engineering environment is instrumented, as appropriate, to automatically record the desired data.
6. Training is provided before installing the technologies for general use.
7. The organization's staff and managers receive feedback on the status and results of the new technologies on a regular basis.
8. Consultation support, appropriate to the expected needs, is established before installing the technologies for general use, and is continued as needed.
9. The specifications for the organization's standard software process and the projects' defined software processes are revised to incorporate the new technologies.

Refer to the organization process definition key process area for practices for revising the organization's standard software process.

Refer to the integrated software management key process area for practices for revising a project's defined software process.

Monitoring implementation

Monitor 1	The cost and effectiveness of initiatives to improve technology are tracked and compared with projected costs and benefits; status and deviations that require management actions are reported to the appropriate managers.
Monitor 2	The cost and schedule performance, quality performance, and process performance of the activities for technology innovation are tracked; status and deviations that require management actions are reported to the appropriate managers.

Verifying implementation

Verification 1	The activities for technology innovation are reviewed with senior management on a regular basis.

These reviews:

1. Summarize the activities for technology innovation.
2. Identify needed strategy changes.
3. Result in the resolution of issues.
4. Result in the approval of revisions to the plans as appropriate.

Verification 2 **The software quality assurance group reviews and audits the activities and products for technology innovation, and reports results as appropriate.**

At a minimum, the reviews and audits cover:

1. The process for preparing the technology innovation plan.
2. The contents of the technology innovation plan.
3. The process for selecting, procuring, and installing new technologies.

Refer to the software quality assurance key process area for practices covering the quality assurance reviews and audits.

Process Change Management

a key process area for Level 5: Optimizing

Process change management involves defining process improvement goals and systematically identifying, evaluating, and implementing improvements to the organization's standard software process and the projects' defined software processes on a continuous basis. Appropriate training and incentive programs are established to allow and encourage all staff and managers to participate in these process improvement activities. Improvement opportunities are identified and evaluated for potential payback for the organization. Pilot efforts are performed to assess new and unproven process changes before they are introduced across the organization.

Goals

Goal 1	The organization's staff and managers are actively involved in setting quantitative, measurable improvement goals and in improving the software process.
Goal 2	The organization's standard software process and the projects' defined software processes continually improve.
Goal 3	The organization's staff and managers are able to use the evolving software processes and their supporting tools and methods properly and effectively.

Commitment to perform

Commitment 1 The organization follows a written policy for implementing software process improvements.

This policy requires that:

1. The organization has quantitative, measurable goals for software process improvement and tracks performance against these goals.
2. The organization's process improvements are directed to improving product quality and increasing productivity.
3. All the organization's staff and managers are expected to participate in improving the software processes.

Skilled and motivated people are recognized as the principle process improvement resource.

4. Comprehensive programs are established to:
 - ☐ improve skills,
 - ☐ enhance job satisfaction, and
 - ☐ ensure appropriate job assignments.

Commitment 2 Senior management oversees the organization's activities for software process improvement.

Specifically, senior management:

1. Establishes the organization's long-term goals and plans for process improvement.
2. Allocates resources for process improvement activities.

- (Commitment 2)**
3. Coordinates with the software managers to ensure they have reasonable and aggressive process improvement goals and effective process improvement plans to meet these goals.
 4. Monitors process improvement performance against goals.
 5. Maintains a consistent priority focus on process improvement in the face of product crises.
 6. Ensures that process improvement issues are promptly resolved.

Ability to perform

Ability 1

An organization-wide program to improve the software process is established which empowers the organization's staff and managers to improve their own working processes and participate in the improvements made by others.

This program includes:

1. A group that manages and supports organizational process improvement.
2. Programs and activities that encourage all employees to submit process improvement proposals which have broad applications.
3. A process for submitting, reviewing, and implementing process improvement proposals.
4. Awards and recognition for employees who originate significant process improvement proposals or large numbers of adopted process improvement proposals.

- (Ability 1) 5. Tools and facilities for process tailoring that support the process improvement activities.

Ability 2 Appropriate resources and funding for software process improvement activities are provided.

1. Resources are allocated to:
 - ☐ lead, guide, and support the process improvement activities;
 - ☐ maintain the process improvement records, files, and database;
 - ☐ develop, control, and disseminate process changes; and
 - ☐ establish and operate the administrative and human resources functions to conduct the communications, motivation, and recognition activities needed to maintain a high level of employee participation.
2. Experienced staff members who have expertise in defining and analyzing software processes are available to help the organization's staff and managers in their process improvement activities.
3. Appropriate tools are made available to the group responsible for coordinating process improvement activities.

Examples of tools include:

- statistical analysis tools,
- database systems, and
- process tailoring tools.

Ability 3 All software managers and senior managers are responsible for the quality and effectiveness of the processes used in their departments.

(Ability 3)

Specifically, they are:

1. Personally involved in communicating the importance of the process improvement activities to all members of their departments.
2. Personally involved in setting process improvement goals and reviewing the process improvement performance of their departments.
3. Committed to supporting the process improvement activities.

Ability 4

The organization's staff and managers receive required training to perform their activities for software process improvement.

1. Software managers and senior software engineering staff are trained in basic process improvement. Training is provided in:
 - ☐ managing technological and organizational change, and
 - ☐ team building and teamwork skills.
2. The software engineering staff is trained in the procedures for proposing process improvements.
3. Senior managers are trained in process improvement:
 - ☐ principles,
 - ☐ goal setting and tracking,
 - ☐ benchmarking and comparative evaluation, and
 - ☐ motivation and team building.

Refer to the training program key process area for practices covering training.

Activities performed

Activity 1

The group responsible for coordinating the organization's software process activities (e.g., software engineering process group) coordinates the process improvement activities.

This group:

1. Defines, establishes, and maintains the process improvement files and database.
2. Defines and maintains the process specifications, procedures, and standard forms for proposing process improvements.
3. Defines goals and measurement plans for overall software process performance (including effectiveness, stability, quality, and productivity).
4. Reviews the process performance goals with senior management for their endorsement.
5. Participates in the effort to define the organization's training needs for process improvement and supports the development and presentation of training course materials.

Refer to the training program key process area for practices covering training.

6. Reviews process improvement proposals and coordinates the actions for these proposals.
7. Tracks status, accomplishments, and participation in the process improvement activities and regularly reports the findings to senior management.

(Activity 1)

8. Coordinates and tracks changes to the organization's and projects' process specifications.

Activity 2

The organization prepares and maintains plans that govern its activities for software process improvement.

These plans:

1. Define the resources required, including staff and tools.
2. Identify the highest priority process areas for improvement.
3. Specify the short-term and long-term goals for process performance and improvement that are measurable and directly related to customer satisfaction indicators.
4. Identify teams and their assignments for addressing improvements for specific process focus areas.

Examples of teams include:

- working groups,
- quality circles, and
- technical committees.

5. Define the procedures for:
 - ☐ the senior managers overseeing the process improvement activities;
 - ☐ the software managers planning and coordinating the process improvement activities;
 - ☐ individuals and teams identifying, evaluating, and introducing appropriate process improvements; and
 - ☐ the teams developing process improvements for specific process focus areas.

(Activity 2)

6. Define the administrative and support plans required to maintain continuous process improvement.
 - ☐ Appropriate administrative procedures are included to facilitate the process improvement process and encourage participation in the process improvement program.
 - ☐ Administrative personnel are included in the process improvement oversight and review activities.
 - ☐ The selection, appraisal, evaluation, and compensation of employees consider their roles and contributions to continuous process improvement.
7. Are reviewed by the plan developers' peers.

Refer to the peer review key process area for practices covering peer reviews.

8. Are reviewed and agreed to by the managers of the software projects before the effort is implemented.

Activity 3

The organization's software process improvement goals and accomplishments reflect the business and strategic operating plans.

Activity 4

Senior management regularly reviews the training, recognition, and motivational programs that support the activities for software process improvement and initiates changes to maintain a high level of employee participation.

Activity 5

Process improvement proposals are submitted by individuals or teams according to a documented procedure.

The process improvement proposals are developed based on:

1. The findings and recommendations of process assessments.

(Activity 5)

2. The organization's stated process improvement goals.
3. Analysis of data on customer problems and customer satisfaction.
4. Analysis of data on organizational performance versus quality and productivity objectives.
5. The results of process benchmarks.
6. The potential for process/task automation.
7. Analysis of data on defect causes.
8. The measured effectiveness of the process activities.
9. Outstanding process improvement proposals.
10. Feedback on previously submitted process improvement proposals.

The process improvement proposals can be submitted at any time and can address any area of the process.

Activity 6

A documented procedure is followed for reviewing, approving, planning, implementing, and tracking process improvement proposals.

This procedure requires that:

1. Each process improvement proposal is evaluated, a decision is made on whether to implement the proposal, and the rationale is documented.
2. The expected benefits of each process improvement are defined.

(Activity 6)

3. The priority of selected process improvement proposals is determined.
4. Implementation of process improvement proposals is assigned and planned.
5. Process improvement efforts that require a substantial effort are assigned to an implementation team.

Examples of substantial efforts include improvements requiring piloting or new technologies and other large changes.

Examples of teams include:

- working groups,
- quality circles, and
- technical committees.

6. The status of the process improvement is tracked.
7. Specific recommendations for changing the organization's standard software process and the projects' defined software processes are documented.
8. Completed process improvement actions are reviewed, verified, and approved before they are closed.
 - ☐ The recommendations are reviewed and approved by the managers of the affected groups and the group responsible for defining and maintaining the affected process specifications.
9. Submitters of process improvement proposals are informed of any decisions regarding their proposals.

Activity 7

Where appropriate, the process improvements are installed on a pilot basis to determine their benefits and effectiveness before they are introduced on a broad scale.

1. Adjustments to the proposed improvement are made and documented during the pilot effort to optimize its implementation.
2. Lessons learned and problems encountered during the effort are documented.
3. The benefits and impacts of the improvement's broader use in the organization are estimated. The uncertainty in these estimates is assessed.
4. A decision is made whether to terminate the effort, proceed with broad-scale implementation of the process improvement, or replan and continue the pilot effort.

Activity 8

Staff and managers actively participate in working groups, quality circles, or technical committees to develop process improvements for assigned process focus areas.

1. Each of these process improvement efforts is funded and the activities are planned and scheduled.
2. Goals are established for each improvement effort; where possible, these goals are defined quantitatively.
3. The plans are approved by the managers of the affected groups and the group that defines and maintains the affected process specifications.

Activity 9

Process change recommendations resulting from completed process improvement actions are incorporated into the organization's standard software process and the projects' defined software processes according to a documented procedure.

(Activity 9)

This procedure requires that:

1. Process changes which are judged to have a major impact on process quality or productivity and process changes which will significantly alter satisfaction of the customer and end users are reviewed and approved by the software task leaders and managers before they are implemented.
2. When decisions are reached to implement the improvement on a broad scale, the changes to the organization's standard software process and the projects' defined software processes, as appropriate, are documented, approved, and disseminated.
 - ☐ Process changes are logged and tracked to ensure that all activities required to complete the changes are performed.
 - ☐ The process changes may be disseminated initially as bulletins or notices rather than updating the software process specifications.
3. Procedures are established and followed to allow ongoing projects to obtain a waiver to delay implementation or not implement individual process improvements.

Refer to the organization process definition for practices covering process waivers.

4. The resources needed to support major process changes are established and funded.
5. The strategy for collecting data to measure and track the process performance changes are documented, reviewed, and agreed to.
 - ☐ This strategy is agreed to by the staff and managers responsible for implementing the software processes affected by the change.
6. The software engineering environment is instrumented, as appropriate, to record the desired data automatically.

(Activity 9)

7. Training courses are updated to reflect the current process, and training is provided before installing the process changes for general use.

Refer to the training program key process area for practices covering training.

8. Consultation support, appropriate to the expected needs, is established before installing the process changes for broad-scale use, and is continued as needed.
9. The process change activity is periodically reviewed to determine when new baseline process specifications should be established and disseminated.
10. The specifications for the organization's standard software process and the projects' defined software processes are revised to incorporate the approved process changes.

Refer to the organization process definition key process area for practices for revising the organization's standard software process.

Refer to the integrated software management key process area for practices for revising a project's defined software process.

Activity 10

A database containing process improvement information is maintained to manage the process improvement proposals.

This database provides the capability to:

1. Ensure that the submitters of the process improvement proposals receive:
 - ☐ prompt acknowledgment of their process improvement proposals,
 - ☐ notification of classification and planned actions for their proposals, and
 - ☐ notification of the disposition of their proposals.
2. Maintain information about the initiation, status, and implementation of process improvement proposals.
3. Ensure focus on high priority process improvement proposals.
4. Ensure focus on process improvement proposals for which the response has been unusually long.
5. Maintain and provide ready access to submitted:
 - ☐ process improvement proposals,
 - ☐ process specifications, and
 - ☐ process changes.
6. Maintain historical data and produce reports on process improvements.

(Activity 10)

Examples of records and reports include those on:

- project productivity, quality, and schedule performance;
- program defect history;
- organizational quality and productivity trends; and
- process development and improvement cost, schedule, and productivity.

7. Show trends in process improvement activities.

Examples of trends include:

- the number of process improvement proposals submitted,
- the time to review and respond to process improvement proposals,
- the percentage of submitted proposals that are accepted,
- the time to implement process changes, and
- participation by the organization's projects, groups, and departments.

Activity 11

The organization's staff and managers receive feedback on the status and results of the process improvement activities on a regular basis.

The feedback provides:

1. A summary of the major improvement activities.
2. Significant innovations and actions taken to address process improvement.
3. A summary status of the process improvement proposals submitted, open, and completed.

(Activity 11)

Examples of means to provide this feedback include:

- electronic bulletin boards,
- newsletters, and
- information flow meetings.

Monitoring implementation

Monitor 1

Measurements are made and used to determine the extent of participation in the process improvement activities.

1. The following are tracked and reported to senior management:
 - ☐ the number of process improvement proposals submitted and implemented for each process area;
 - ☐ the number of process improvement proposals submitted by each of the projects, groups, and departments;
 - ☐ the number and types of awards and recognitions received by each of the projects, groups, and departments;
 - ☐ the response time for handling process improvement proposals (senior management is notified when response time exceeds a predefined limit);
 - ☐ the percentage of process improvement proposals accepted per reporting period; and
 - ☐ the overall change activity, including number, type, and size of changes.
2. The effect of implementing each process improvement is compared to its defined goals; results are used to identify changes to the process improvement process.

- Monitor 2** **Measurements are made and used to determine how effectively the process improvement activities are focused on and relate to customer needs and customer satisfaction.**
1. Process measurements are defined that relate to indicators of the customer's satisfaction.
 2. External product customers and internal process customers are identified.
 3. Customer needs and the degree to which the (internal and external) customer is satisfied with the supplier's performance are determined.
- Monitor 3** **Measurements are made and used to determine how effective the process improvement activities are overall.**
1. Overall performance of the organization's and projects' processes, including effectiveness, quality, and productivity, are tracked over time against defined goals; results are reported to senior management.
 2. Overall productivity and quality trends for each project are tracked over time; results are reported to the project managers and to senior management.
- Monitor 4** **Process aging measurements (e.g., time since the process specification was last updated) are maintained to identify process activities that are not being improved.**
1. Process specifications which have not been updated in a predefined long period (e.g., 36 months) are flagged for priority attention.
 2. Process specifications which have not been updated in a predefined short period (e.g., 18 months) are flagged for investigation.

Verifying implementation

Verification 1 **The activities for software process improvement are reviewed with the project managers and the senior managers on a regular basis.**

These reviews are held to summarize participation in the process improvement activities, assess process performance, identify needed goal changes, resolve issues, and approve revised plans as appropriate.

Verification 2 **The software quality assurance group reviews and audits the activities and products for process improvement, and reports results as appropriate.**

At a minimum, the reviews and audits cover:

1. The preparation of the organization's software process improvement plan.
2. The process of initiating, submitting, reviewing, approving, and planning implementation of process improvement proposals.
3. The degree to which the process measurements conform to specifications and reflect actual performance.
4. The process for documenting, reviewing, approving, controlling, and disseminating changes to the software processes.
5. The degree to which process improvement activities are consistently measured and tracked.
6. The degree to which actual process improvement performance achieves the plans and goals.

(Verification 2)

Refer to the software quality assurance key process area for practices covering the quality assurance reviews and audits.

Appendix A: References

- IEEE91 The Institute of Electrical and Electronic Engineers, Inc., IEEE Std 610.12.1990, "Glossary of Software Engineering Terminology (ANSI)," *Software Engineering Standards Collection*, Wiley-Interscience, New York, NY, 1991.
- Fowler90 P. Fowler and S. Rifkin, "Software Engineering Process Group Guide," Software Engineering Institute, CMU/SEI-90-TR-24, September 1990.
- Hauser88 J.R. Hauser and D. Clausing, "The House of Quality," *Harvard Business Review*, May-June 1988, pp. 63-73.
- Humphrey87 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.
- Humphrey89 W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, Reading, MA, 1989.
- Kezsbom89 D.S. Kezsbom, D.L. Schilling, and K.A. Edward, *Dynamic Project Management*, John Wiley & Sons, New York, NY, 1989.

References

- Masaaki86 I. Masaaki, *Kaizen: The Key to Japan's Competitive Success*, McGraw Hill, New York, NY, 1986.
- Mays90 R.G. Mays, C.L. Jones, G.J. Holloway, and D.P. Studinski, "Experiences With Defect Prevention," *IBM Systems Journal*, Vol. 29, No. 1, 1988, pp. 4-32.
- Pall87 G.A. Pall, *Quality Process Management*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.
- Radice85 R.A. Radice, N.K. Roth, A.C. O'Hara, Jr., and W.A. Ciarfella, "A Programming Process Architecture," *IBM Systems Journal*, Vol. 24, No. 2, 1985.
- Paulk91 M.C. Paulk, B. Curtis, M.B. Chrissis, et. al., "Capability Maturity Model for Software," Software Engineering Institute. CMU/SEI-91-TR-24, August 1991.
- Taguchi90 G. Taguchi and D. Clausing, "Robust Quality," *Harvard Business Review*, January-February 1990, pp. 65-75.

Appendix B: Glossary of Terms

Definitions Of Terms

ability to perform - (See *common features*.)

acceptance criteria - The criteria that a system or component must satisfy in order to be accepted by a user or customer. [IEEE91]

acceptance testing - Formal testing conducted to determine whether or not a system or product satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system or product. [IEEE91]

activity - Any step taken or function performed, both mental and physical, toward achieving some objective.

activities performed - (See *common features*.)

allocated requirements - (See *system requirements allocated to software*.)

application domain - A bounded set of related systems (i.e., systems that address a particular type of problem) which usually require special skills or resources to be developed. Examples include payroll and personnel systems, command and control systems, compilers, and expert systems.

audit - An independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria. [IEEE91]

baseline - A specification or product that has been formally reviewed and agreed upon, which thereafter serves as the basis for future development, and which can be changed only through formal change control procedures. [IEEE91]

baseline item - A specific process specification or product that has been formally reviewed and agreed upon, which thereafter serves as the basis for future development, and which can be changed only through formal change control procedures. (See *baseline process* and *baseline product*.)

Glossary of Terms

baseline process - A specific process specification that has been formally reviewed and agreed upon, which thereafter serves as the basis for future development, and which can be changed only through formal change control procedures.

baseline product - A specific product that has been formally reviewed and agreed upon, which thereafter serves as the basis for future development, and which can be changed only through formal change control procedures.

bidder - An individual, partnership, corporation, or association who has submitted a proposal and is a candidate to be awarded the contract to design, develop, and/or manufacture items.

benchmark - A standard against which measurements or comparisons can be made. [IEEE91]

capability maturity model - A model of the stages through which software organizations progress as they define, implement, evolve, and improve their software process. This model provides a guide for selecting process improvement strategies by determining current process capabilities and identifying the issues most critical to software quality and process improvement.

causal analysis - The analysis of defects to determine their underlying root cause.

causal analysis meeting - A meeting, conducted after completing a specific task, to analyze defects uncovered during the performance of that task.

commitment - A pact which is freely assumed, visible, and expected to be kept by all parties.

commitment review procedure - A procedure that has the following elements: (1) the affected parties make commitments with their full knowledge and agreement, (2) the affected parties meet on a regular basis to jointly review commitments, (3) actual and potential problems in meeting commitments are identified, and issues are escalated up the management chain until they are adequately addressed, (4) changes to commitments are

approved by all affected parties, (5) changes to commitments are reflected in the appropriate plan documents only when they are approved, and (6) approved changes to planned commitments are communicated to all affected parties.

commitment to perform - (See *common features*.)

common cause (of a defect) - A cause of a defect that is inherently a result of the overall operating environment. (See *special cause* for contrast.)

common features - The subdivision categories of the CMM key process areas. The CMM common features are the following:

- ❑ *commitment to perform* - The actions taken by the organization that ensure the process is established and will endure. The documented policies, procedures, charters, and other rules and principles adopted by the organization to influence and determine decisions are a significant contributor to the organization's performance.
- ❑ *ability to perform* - The preconditions that must exist in the project and/or organization to implement the process competently. It includes staff resources and skills, special tools, assignment of critical project responsibilities, and establishment of essential training to support these needs.
- ❑ *activities performed* - The process steps that must be performed to effectively implement the key process area. It includes the plans, standards, procedures, methods, and guidelines that describe how the process is normally carried out. It also includes process steps for any corrective action that needs to be taken.
- ❑ *monitoring implementation* - The process steps that must be performed to take measurements, analyze measurements, and take action based on the results in order to determine the status of the process, and control, correct, and improve it (i.e., the practices of that key process area).
- ❑ *verifying implementation* - The process steps that must be performed for observing and checking the process activities to guide and ensure that these activities are performed and comply with the process definition.

Glossary of Terms

configuration - The functional and physical characteristics of hardware or software as set forth in technical documentation or achieved in a product. [IEEE91]

configuration component - The lowest level entity (of a configuration item) that can be placed into, and retrieved from, a configuration management library.

configuration control - An element of configuration management, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification. [IEEE91]

configuration control board - (See *software configuration control board*.)

configuration-controlled product - (See *configuration item*.)

configuration item - An aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process. [IEEE91]

configuration management - The process of identifying and defining the configuration items in a system, controlling the release and change of these items throughout the system life cycle, recording and reporting the status of configuration items and change requests, and verifying the completeness and correctness of configuration items. [IEEE91]

contingency factor - An adjustment (increase) of a size, cost, or schedule plan to account for likely under-estimates of these parameters due to incomplete specification, inexperience in estimating the application domain, etc.

contract data requirements list - A list that identifies all data items that a contractor is required to deliver to its customer. For each data item the list specifies the delivery requirements (such as standards it must satisfy) due dates, and customer approval procedures.

Glossary of Terms

contract terms and conditions - The stated legal, financial, and administrative aspects of a contract.

control limits - (on a statistical control chart) Values established above and below an expected value. These limits are used as a guide to help determine which measured results are likely to be caused by random variation and which ones are likely to represent unusual behavior.

critical computer resource - The parameters of the computing resources deemed to be a source of risk to the project because the potential need for those resources may exceed the amount that is available.

customer - The person or organization that is responsible for accepting the contracted product and authorizing payment to the developing organization.

defect - Any unintended characteristic that impairs the utility or worth of an item.

defect density - The number of defects identified in a product divided by the size of the product component (expressed in standard measurement terms for that product).

defect prevention - The activities involved in identifying defects and preventing them from being introduced into a product.

defect root cause - The underlying reason (e.g., process deficiency) that allowed a defect to be introduced.

defined software process - A well-characterized, documented, and understood software process, defined in terms of software standards, procedures, tools, and methods. A defined process is typically needed to control, coordinate, and improve the activities of numbers of people in complex tasks.

dependency - (See *dependency item*.)

Glossary of Terms

dependency item - A product, an action, a piece of information, etc. that must be provided by one person or group to a second person or group so that the second person or group can perform a planned task.

documented procedure - A reasonably complete set of rules and steps for performing a task. The rules and steps are written down and can be used by individuals who are knowledgeable about the task to perform the task in a repeatable way.

end user - The person or persons who will use the system for its intended operational use when it is deployed in its environment.

findings - The conclusions of an assessment, evaluation, audit, or review that identify the most important issues, problems, or opportunities within the area of investigation.

first-line software manager - A manager who has direct management responsibility (including providing technical direction and administering the personnel and salary functions) for the staff and activities of a single department of software engineers and other related staff.

formal configuration management - Defining baselined configuration items and procedurally controlling changes to baselines. Formal configuration management can be contrasted with the developmental configuration management, where the definition and change control of items may be performed informally.

formal review - A formal meeting at which a product is presented to the end user, customer, or other interested parties for comment and approval. It can also be a review of the management and technical activities and progress of the hardware/software development project.

function - A set of related actions, taken by individuals, tools, etc. specifically assigned or fitted for their roles, to accomplish a definite purpose or end.

goals - A summary description of the key practices in the key process area that can be used to test alternative implementations of the key practices to

Glossary of Terms

determine if they satisfy the intent of the key process area. The goals signify the scope and boundaries of the key process area.

group - The collection of organizationally-related departments, managers, and staff who have responsibility for a set of project activities performed in a specific discipline (e.g., hardware engineering, software engineering, legal). A group could be as small as a single person dedicated part-time within a department or could be multiple departments, depending on the size of the project and other parameters.

host computer - The computer on which a program or other software component resides, particularly a computer used for software development or to emulate the target computer. (See *target computer* for contrast.)

institutionalize - The incorporation of the methods, practices, and procedures, used to perform a specific function, as a permanent part of an organization's culture and way of doing business.

key indicators - The key practices, or components of a key practice, that have been identified as providing the greatest insight into whether the goals of a key process area are satisfied.

key practices - The policies, infrastructures, and activities that contribute most to the implementation and institutionalization of a key process area.

key process area - A cluster of related activities that, when performed collectively, achieve a set of goals important for enhancing process capability. The software process areas identified by the SEI to be the principal building blocks to help recognize the software process capability of an organization and understand the improvements needed to advance to higher maturity levels. A subdivision of the capability maturity model for software.

life cycle - (See *software life cycle*.)

life-cycle model - (See *software life-cycle model*.)

Glossary of Terms

maturity level - A well-defined development plateau toward achieving a mature software process.

maturity questionnaire - A set of yes/no questions about the software process that sample the key practices in each key process area of the CMM. It is used as a springboard to measure the capability of an organization or project to reliably execute the software process.

measurement - The dimension, capacity, quantity, or amount of something, (such as 300 source lines of code or 7 document pages of design). (See *metric* for comparison.)

method - A reasonably complete set of rules that establishes a precise and repeatable way of performing a task and arriving at a desired result.

methodology - A collection of methods, procedures, and standards that defines an integrated synthesis of engineering approaches to the development of a product.

metric - A unit of measurement (such as source lines of code or document pages of design). (See *measure* for comparison.)

mid-level software manager - A manager who reports directly or indirectly (i.e., through another manager) to the project software manager and who has direct management responsibility (including providing technical and management direction, and administering the personnel and salary functions) for other software managers.

milestone - A scheduled event for which some project member or manager is held accountable and that is used to measure progress.

monitoring implementation - (See *common features*.)

operational software - The software that is intended to be used and operated in the system when it is deployed in its intended environment.

Glossary of Terms

organization - A unit within a company, agency, or service that shares common management, is centered at a single geographical site, and has responsibility for a common business area.

organization's software process assets - A set of process artifacts, maintained by an organization, for use by the projects in defining, maintaining, and implementing their software process. These software process assets include a standard software process, the guidelines and criteria for tailoring the organization's standard software process, software life-cycle models approved for use by the projects, a library of process specifications previously developed by projects in the organization, and a software process database for the organization.

organization's standard software process - (See *standard software process*.)

Pareto analysis - (for analyzing defects) The analysis of defects in which a Pareto graph (i.e., a bar graph of the number of defects by category found in a specified time period) is created and analyzed to determine which categories of defects should be addressed to produce the biggest improvement. Pareto analysis is based on the generalization that 80% of the defects are in 20% of the products, the so-called 80/20 rule.

peer review - A review of a software product, following defined procedures, by peers of the producer(s) of the product for the purpose of identifying defects and improvements.

peer review leader - A person specifically trained and qualified to plan, organize, and lead a peer review.

periodic review/activity - A review/activity that occurs at a specified regular time interval, rather than at the completion of major events.

policy - A guiding principle, typically established by senior management, adopted by an organization to influence and determine decisions.

post-mortem meeting - A meeting held after a well-defined activity or set of related activities is completed. The purpose of the meeting is to review the

Glossary of Terms

completed activities, review the results, and determine and document problems encountered and lessons learned.

practices - Requirements employed to prescribe a disciplined uniform approach to a process. [IEEE91]

prime contractor - An individual, partnership, corporation, or association who administers a subcontract to design, develop, and/or manufacture items.

procedure - A written description of a course of action to be taken to perform a given task. [IEEE91]

process - A series of actions, changes, or functions that achieve an end or result.

process aging measure - The elapsed time since the specification of the process was revised.

process architecture - (See *software process architecture*.)

process capability - The range of expected results that can be achieved by following a process (See *process performance* for comparison.)

process database - (See *software process database*.)

process design - (See *software process design*.)

process kernel - (See *software process kernel*.)

process maturity - The extent to which a specific process is explicitly defined, managed, measured, controlled, and effective.

process measurement - The set of definitions, methods, and activities used to take measurements of a process and the resulting products for the purpose of characterizing and understanding the process.

Glossary of Terms

process performance - A measure of the actual results achieved by following a process. (See *process capability* for comparison.)

process performance baseline - A documented characterization or model of a specified process which is used as a benchmark for comparing actual process performance against the expected process performance.

process specification - A document that specifies, in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristics of a process; it also may include the procedures for determining whether these provisions have been satisfied.

process tailoring - The activity of creating a process specification by elaborating and completing the details of process kernels or other incomplete specifications of a process.

product - (See *software product* and *software work product*.)

profile - A comparison, usually in graphical form, of plans or projections versus actuals over time.

project - An effort focused on a specific product, which may include hardware as well as software.

project engineering plan - A document used to plan, coordinate, and track the project engineering activities that affect, or are of concern to, multiple engineering groups.

project manager - The person who has total business responsibility for the entire project; the person who directs, controls, administers, and regulates a project building a software or hardware/software system. The project manager is the person ultimately responsible to the customer.

project software manager - A manager who has total responsibility for all the software activities for the entire project. The project software manager is the person the project manager deals with in terms of software commitments, etc., and the person who controls all the software development staff for a project.

Glossary of Terms

quality assurance - (See *software quality assurance*.)

quality control - A set of activities designed to evaluate the quality of developed products. [IEEE91]

regular review/activity - A review/activity that occurs at intervals determined by the occurrence of recurring major events.

reserve capacity - That quantity of an available computer resource (e.g., memory) that is not allocated for use by any of the software functions.

review leader - (See *peer review leader*.)

schedule critical path - A series of dependent tasks for a project that must be completed as planned in order to keep the entire project on schedule.

senior manager - A manager at a high enough level that his/her primary focus would be expected to be the long-term vitality of the company and organization, rather than short-term project and contractual concerns and pressures. In general, a senior manager for engineering would have responsibility for multiple projects.

software baseline - A set of configuration items (software documents and software components) that has been formally reviewed and agreed upon, that thereafter serves as the basis for future development, and that can be changed only through formal change control procedures.

software baseline audit - A process of reviewing the structure, contents, and facilities of the configuration management library system and the software configuration management process that is followed to assess the integrity of the software baseline.

software build - (noun) An operational version of a software system or component that incorporates a specified subset of the capabilities the final software system or component will provide. [IEEE91]

Glossary of Terms

software build - (verb) A process of putting together selected software components to provide the set or specified subset of the capabilities the final software system will provide.

software capability evaluation - An appraisal by a trained team of professionals, using a method such as the SEI software capability evaluation method, to (1) identify contractors who are qualified to perform the software work, or (2) monitor the state of the software process used on an existing software effort.

software configuration control board - A group of people responsible for and empowered by the project software manager to evaluate and make the decisions that affect the software baseline.

software development plan - The collection of plans that describe the activities to be performed for the software project. It governs the management of the activities performed by the software engineering group for a software project. It includes the software development plan (as in DOD-STD-2167A), but it is not limited to the scope of any particular planning standard such as DOD-STD-2167A and IEEE-Std-1058, which may use similar terminology.

software development process - The set of activities, methods, and practices that guide people in the creation and production of a set of software artifacts.

software engineering capability - The ability of a software organization to perform successfully (in terms of cost, schedule, product functionality, and quality). The capability has several dimensions, including (1) the expertise, experience, training, and motivation of the people performing and managing the work, (2) the process capability, and (3) the technology that is available and applied.

software engineering group - The collection of departments, managers, and staff who have responsibility for the software development activities (i.e., requirements analysis, design, code, and test) covered by the CMM key process areas for the project. Groups such as the software quality assurance group, the software configuration management group, and the software

Glossary of Terms

engineering process group are not included in the software engineering group.

software engineering process group - A group of specialists who facilitate the definition and improvement of the software process used by the organization.

software engineering staff - The software technical people (e.g., analysts, programmers, and engineers), including software task leaders, who are not managers and who perform the software development activities covered by the CMM key process areas (i.e., software requirements analysis and specification, software design, software coding, and software testing).

software life cycle - The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and, sometimes, retirement phase. [IEEE91]

software life-cycle model - A characterization of a software life cycle that specifies the primary stages of the life cycle and the relationships between these stages.

software manager - Any manager, at any management level, who has direct responsibility for any portion of the software for a project.

software process - 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).

software process action plan - A plan, derived from the recommendations of a software process assessment, that identifies the specific actions that will be taken to improve the software process and outlines the plans for implementing those actions.

software process architecture - A conceptual framework for incorporating software process tasks and activities in consistent ways. It describes and orders the software tasks and activities that are common across the organization's projects and identifies the major software process components and their connections.

software process assessment - An appraisal by a trained team of software professionals, using a method such as the SEI assessment method, 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 start the actions needed for software process improvement.

software process assets - (See *organization's software process assets*)

software process capability - (See *process capability*.)

software process database - A managed and controlled database that serves as an organizational repository and contains all the critical information on the implementation of the software processes.

software process design - A high-level (summary) specification of a project's software process for the selected software life-cycle model. It describes the software tasks and activities that are performed and identifies the major software process components and their connections.

software process kernel - A process component of an organization's standard software process that is used by a project as the basis for developing a process specification. Software process kernels contain the core requirements of the major software process components identified in the software process architecture.

software process maturity - (See *process maturity*.)

software process performance - (See *process performance*.)

software process specification - The operational definitions of the major software process components identified in the project's defined software

Glossary of Terms

process. A document that specifies the requirements, design, behavior, or other characteristics of a software process in a complete, precise, verifiable manner; it may also include the procedures for determining whether these provisions have been satisfied. (See *process specification*.)

software product - The complete set, or any of the individual items of the set, of computer programs, procedures, and associated documentation and data designated for delivery to a customer or end user. [IEEE91] (See *software work product* for comparison.)

software project - The effort focused on specifying, designing, developing, and testing the software components of a project.

software quality assurance - A planned and systematic means for assuring management that defined standards, practices, procedures, and methods of the software process are applied.

software quality plan - A plan that documents the quality actions that a project intends to implement to address the customer's and end-users' product quality needs and desires.

software task leader - (See *task leader*.)

software work product - Any artifact created as part of the software process, including computer programs, plans, procedures, and associated documentation and data, that may not be intended for delivery to a customer or end user. (See *software product* for comparison.)

special cause (of a defect) - A cause of a defect that is specific to some transient circumstances (such as a specific local condition, a single machine, a single individual, or a small group of people) performing in an unexpected way. (See *common cause* for contrast.)

staff - The people, including task leaders, who are not managers and who are responsible for accomplishing the assigned business function

stage - A partition of the software effort that is of a manageable size and that represents a meaningful and measurable set of related tasks which are performed by the project.

stage review - A review conducted at the start and end of a stage to determine if the stage readiness criteria or completion criteria are satisfied.

standard - An approved, documented, and available set of criteria used to determine the adequacy of an action or object.

standard software process - A process, specified at the organizational level, that includes a process architecture of a start-to-end software process and the specifications of software process kernels for the processes that are common across the organization's projects.

statement of work - In system/software projects, a description of all the work required to complete a project.

statistical process control - The use of statistical techniques to analyze a process through its various outputs so as to take appropriate actions toward maintaining or improving the process. [Pall87]

subcontract manager - A prime contractor manager who has direct responsibility for administering and managing a subcontract.

subcontractor - An individual, partnership, corporation, or association who contracts with an organization to design, develop, and/or manufacture items.

subprocess - A part of a process that has defined inputs and preconditions, and which, when followed, brings about a particular result.

system - A collection of components organized to accomplish a specific function or set of functions. [IEEE91] An assembly of things or parts forming a complex or unitary whole.

Glossary of Terms

system design review team - A group of engineers who represent all engineering groups of a project and who are responsible for ensuring the overall integrity and quality of the system and all its components.

system design team - A designated group of people responsible for designing a hardware-software system.

system engineering group - The collection of departments, managers, and staff who have responsibility for specifying the system requirements, allocating the system requirements to the hardware and software components, specifying the interfaces between the hardware and software components, and monitoring the design and development of these components to ensure conformance with their specifications.

system requirements allocated to software - The subset of the system requirements that are to be implemented in the software components of the system. The elaboration and refinement of the allocated requirements result in the software requirements specification and are a primary input to the software development plan.

tailor - To adapt a set of standards or procedures to better match process or product requirements.

target computer - The computer on which delivered software is intended to operate. (See *host computer* for contrast.)

task - A unit of work in the software process, performed by a team of people, that provides a visible management checkpoint. Tasks have readiness criteria (preconditions) and completion criteria (postconditions).

task kickoff meeting - A meeting held at the beginning of a task of a project for the purpose of preparing the individuals involved to effectively perform the activities of that task.

task leader - A technical team leader for a specific task, who has technical responsibility and provides the technical direction to the staff (including him/herself) working on that task. Usually the task leader will work for the same first-line manager as the other people working on that activity.

Glossary of Terms

team - A collection of people, often drawn from diverse but related groups, to perform a well-defined function for an organization or a project. Team members are often part-time participants of the team and have other primary responsibilities.

technology - The tools and methods that can be applied by people in accomplishing some particular result.

time phasing - The ordering of tasks on a time scale to satisfy some criteria, such as balancing workload and accommodating task dependencies.

traceability - The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another. [IEEE91]

unprecedented task - A task that is significantly different (e.g., in complexity or in technology) from any task the organization has worked on previously.

verifying implementation - (See *common features*.)

Glossary of Terms

Appendix C: Abridged Version of the Key Practices

This appendix provides an abridged version of the key practices, which provides a high-level overview of the primary activities the SEI prescribes for each key process area. It can be used to get a "quick look" at each key process area. It does not, however, provide the specific activities for these key practices nor does it cover all the key practices. It is intended for informational purposes, not for determining compliance to the key practices or planning process improvements.

In this abridgement, each key process area is shown on two pages. The first page provides a short descriptive statement of the key process area and a statement of its goals. The second page lists the top-level key practice statements from the "activities performed" common feature of the key process area. These items are extracted verbatim from the detailed key practice tables.

There are a number of other key practices specified under the other common features (i.e., commitment to perform, ability to perform, monitoring implementation, and verifying implementation) that are not contained in this appendix. These other key practices must be in place to ensure the key practices are implemented appropriately and effectively, are solidly established, will be maintained and not erode over time, and can be effectively applied to new work. To appropriately establish a key process area, the full set of key practices should be used.

Level 2: Requirements Management

Requirements management involves establishing and maintaining an understanding and agreement with the customer on the requirements for the software throughout the software life cycle. The agreements cover both the technical requirements for the software and the nontechnical requirements, such as delivery dates for the software. The agreements form the basis for estimating, planning, performing, and tracking the project's software activities.

The goals of requirements management are:

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

The top-level activities performed for requirements management are:

1. The allocated requirements are documented in a consistent format and are clearly stated, verifiable, and testable.
2. The software engineering group reviews and agrees to the allocated requirements before they are incorporated into the software efforts.
3. The allocated requirements form the basis for the software plans, products, and activities.
4. Changes to the allocated requirements are appropriately reviewed and incorporated into the software efforts.

Level 2: Software Project Planning

Software project planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work. A plan is established to address the commitments to the customer according to the resources, constraints, and capabilities of the project. The plan provides the basis for initiating the software effort and managing the progress of the work.

The goals of software project planning are:

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

The top-level activities performed for software project planning are:

1. The software engineering group is an active participant on the project proposal team.
2. The software planning is initiated in the early stages of, and in parallel with, the overall project planning.
3. The software engineering group actively participates in the overall project planning throughout the project's life.
4. Senior management reviews and approves all commitments made to individuals and groups external to the organization.
5. A software life-cycle model with predefined stages of manageable size is identified or defined.
6. The project's software development plan is developed according to a documented procedure.
7. The software development plan covers (directly or by reference) the plan for the software activities.
8. Software products and software process specifications that are needed to establish and maintain stability of the software activities are explicitly identified to be controlled project baseline items.
9. Estimates for the size of the software products are derived according to a documented procedure.
10. Estimates for software development resources and costs are derived according to a documented procedure.
11. Estimates for critical target computer resources are derived according to a documented procedure.
12. The project's software schedule is derived according to a documented procedure.
13. The software technical, cost, resource, and schedule risks are identified, assessed, and documented.
14. Plans for the project's software engineering facilities, environments, and support tools are prepared.
15. Software planning data are recorded for use by the project.

Level 2: Software Project Tracking and Oversight

Software project tracking and oversight involves tracking and reviewing the software accomplishments and results against documented estimates, commitments, and plans, and adjusting these based on the actual accomplishments and results. A documented plan for the software effort is used as the basis for tracking the software activities, communicating status, and revising plans. The software activities are monitored by the software managers on a regular basis. Regular technical reviews and reviews with the project manager and senior management are conducted to ensure that management and staff are aware of the software status and plans, and that issues receive appropriate attention.

The goals of software project tracking and oversight are:

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

The top-level activities performed for software project tracking and oversight are:

1. A documented software development plan is used for tracking the software activities and communicating status.
2. Senior management reviews and approves all commitments and commitment changes made to individuals and groups external to the organization.
3. Approved changes to software commitments or commitments affecting the software activities are explicitly communicated to the staff and managers of the software engineering group and software-related groups.
4. The project's software size is tracked and corrective actions are taken.
5. The project's software costs are tracked and corrective actions are taken.
6. The project's critical target computer resources are tracked and corrective actions are taken.
7. The project's software schedule is tracked and corrective actions are taken.
8. Software engineering technical activities are tracked and corrective actions are taken.
9. The software technical, cost, resource, and schedule risks are tracked throughout the life of the project.
10. Actual measured data and replanning data for the software project tracking activities are recorded for use by the software engineering staff and managers.
11. The software engineering staff and managers conduct regular reviews to track technical progress, plans, performance, and issues against the software development plan.
12. Formal reviews, to address the accomplishments and results of project software engineering, are conducted at selected project milestones and at the beginning and completion of selected stages.

Level 2: Software Subcontract Management

Software subcontract management involves selecting a software subcontractor, establishing commitments with the subcontractor on the work to be performed, coordinating activities with the subcontractor, and tracking and reviewing the subcontractor's performance and results. The subcontractor is selected based on its ability to perform the work. A documented agreement covering the technical and nontechnical (e.g., legal, financial, and administrative) requirements is established and is the basis for managing the subcontract. Regular technical and management reviews are conducted to ensure that management and staff of both organizations are aware of the software status and plans, and that issues receive appropriate attention.

The goals of software subcontract management are:

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

The top-level activities performed for software subcontract management are:

1. The work to be subcontracted is defined and planned according to a documented procedure.
2. The software subcontractor is selected according to a documented procedure, based on a complete evaluation of the subcontract bidders' ability to perform the work.
3. The contractual agreement between the prime contractor and the subcontractor establishes the basis for managing the subcontract.
4. A documented subcontractor's software development plan, which covers (directly or by reference) the appropriate items from the prime contractor's software development plan, is reviewed and approved by the prime contractor.
5. A documented and approved subcontractor's software development plan is used for tracking the software activities and communicating status.
6. Changes to the subcontracted scope of work, subcontract terms and conditions, and other commitments are resolved according to a documented commitment review procedure involving affected groups of both the prime contractor and the subcontractor.
7. The prime contractor's management conducts regular status/coordination reviews with the subcontractor's management.
8. Periodic technical reviews and interchanges are held with the subcontractor.
9. Formal reviews to address the subcontractor's software engineering accomplishments and results are conducted at selected milestones and at the completion of selected stages.
10. The prime contractor's software quality assurance group monitors the *subcontractor's software quality assurance activities according to a documented procedure.*
11. The prime contractor's software configuration management group monitors the subcontractor's activities for software configuration management according to a documented procedure.
12. The prime contractor conducts acceptance testing as part of the delivery of the subcontractor's products according to a documented procedure.
13. The subcontractor's performance is evaluated on a periodic basis and the evaluation is reviewed with the subcontractor.

Level 2: Software Quality Assurance

Software quality assurance involves reviewing and auditing the software products and activities to ensure that they comply with the applicable processes, standards, and procedures, and providing the staff and managers with the results of their reviews and audits. The software quality assurance function is required on all projects. The group performing this function is independent of the software groups and project management. A senior manager who is committed to handling all major software quality assurance issues is identified. Where compliance issues exist, the software quality assurance group works with the appropriate managers, including senior management where required, to resolve the issues.

The goals of software quality assurance are:

1. Compliance of the software product and software process with applicable standards, procedures, and product requirements is independently confirmed.
2. When there are compliance problems, management is aware of them.
3. Senior management addresses noncompliance issues.

The top-level activities performed for software quality assurance are:

1. A SQA plan is prepared for each software project according to a documented procedure.
2. The SQA activities are performed in accordance with the SQA plan.
3. The SQA group participates in the preparation, review, and approval of the project's software development plan, process specifications, standards, and procedures.
4. The SQA group reviews and audits the software engineering activities to ensure process compliance.
5. The SQA group reviews representative samples of deliverable and designated nondeliverable software products to ensure compliance with the designated process requirements.
6. The SQA group regularly reports the results of its reviews and audits to the software engineering staff and managers.
7. Deviations identified in the software engineering activities are documented and handled according to a documented procedure.
8. The SQA group conducts regular reviews of its activities and findings with the customer's SQA personnel.

Level 2: Software Configuration Management

Software configuration management involves selecting project baseline items (e.g., the project description, products, and process specifications of the project), controlling these items and changes to them, and recording and reporting status and change activity for these items. Changes to these baseline items are controlled systematically using a defined change control process. The configuration (software and documentation) of a system, or of any of the controlled intermediate or support products, can be distinctly identified at any point in time.

The goals of software configuration management are:

1. Controlled and stable baselines are established for planning, managing, and building the system.
2. The integrity of the system's configuration is controlled over time.
3. The status and content of the software baselines are known.

The top-level activities performed for software configuration management are:

1. Different levels of SCM are implemented, as appropriate, during the project's life cycle.
2. A documented SCM plan exists.
3. A documented and approved SCM plan is used as the basis for performing the SCM activities.
4. A configuration management library system is established as a repository for the software baselines.
5. The software engineering products and process specifications (i.e., configuration items) to be placed under configuration management are identified.
6. A documented procedure is followed for initiating, recording, reviewing, approving, and tracking change requests and trouble reports for all configuration items.
7. A documented procedure is followed to control changes to configuration items.
8. A documented procedure is followed to create and control the release of software baseline products.
9. A documented procedure is followed to record the status of configuration items and change requests.
10. Standard reports documenting the SCM activities and the contents of the software baseline are created and distributed to affected groups and individuals.
11. A documented procedure is followed to prepare for, conduct, report results from, and track action from software baseline audits.

Level 3: Organization Process Focus

Organization process focus involves developing and maintaining an understanding of the organization's software processes and coordinating the activities to specify and improve these processes. A group such as a software engineering process group acts as the focus for the software process activities in the organization. This group coordinates the projects' software process definition activities and the organization's long-term process improvement efforts.

The goals of organization process focus are:

1. Current strengths and weaknesses of the organization's software process are understood and plans are established to systematically address the weaknesses.
2. A group is established with appropriate knowledge, skills, and resources to define a standard software process for the organization.
3. 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.

The top-level activities performed for organization process focus are:

1. The software process is assessed periodically, and action plans are developed to address the assessment findings.
2. The organization follows a documented and approved plan to coordinate its activities for software process definition and improvement.
3. The organization's and projects' activities for defining, implementing, measuring, and improving the software process are coordinated at the organization level.
4. The use of the software process database for the organization is coordinated at the organizational level.
5. New technologies in limited use in the organization are monitored, evaluated, and, where appropriate, implemented in other parts of the organization.
6. Training for the organization's and projects' software process is coordinated across the organization.
7. The staff and managers of the groups involved in implementing the software processes participate in, and are informed of, the organization's and projects' activities for software process definition and improvement.

Level 3: Organization Process Definition

Organization process definition involves establishing and maintaining a standard software process for the organization, along with related items, for use by the projects in establishing their software process. This standard software process provides a common process for software development and software maintenance projects. It defines the essential process steps for the projects and establishes the common basis for measuring process performance across all projects and for long-term improvement.

The goals of organization process definition are:

1. A standard software process for the organization is defined and maintained as a basis for stabilizing, analyzing, and improving the performance of the software projects.
2. Specifications of common software processes and documented process experiences from past and current projects are collected and available.

The top-level activities performed for organization process definition are:

1. The organization develops and maintains a repository of software process assets for use by the projects.
2. The organization's standard software process is developed according to a documented procedure.
3. The specifications of the organization's software process kernels are documented according to established organization standards.
4. Software life-cycle models that are approved for use by the projects are identified and documented.
5. A library of software process specifications previously developed by projects in the organization is established and used.
6. A software process database for the organization is established and used according to a documented procedure.
7. The organization's software process assets are revised according to a documented procedure.
8. Large efforts to develop or revise the organization's software process assets are planned, and the plan is documented, reviewed, and approved.
9. The group responsible for coordinating the organization's software process activities (e.g., software engineering process group) reviews changes to all projects' defined software processes that conflict with, or would improve, the organization's standard software process.

Level 3: Training Program

Training program involves identifying the training needs of the organization, the projects, and the individuals and developing and procuring training courses to address these needs. The training program ensures that training needed to perform each of the organization's job functions is appropriate and is not circumvented inappropriately.

The goals of training program are:

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

The top-level activities performed for training program are:

1. Each project develops and maintains a training plan specifying its training needs.
2. The organization's training plan is developed and revised periodically.
3. The organization's training program is developed and revised periodically according to a documented procedure.
4. A waiver procedure for required training is established and used to independently determine whether an individual possesses the knowledge and skills covered in the training course.
5. Where appropriate, training courses are developed, maintained, and conducted at the project level.
6. Training courses developed at the organization level are developed and maintained according to organization standards.
7. Where appropriate, training sessions for individuals are tied to their job responsibilities so that on-the-job activities or other outside experiences will reinforce the training within a reasonable time after the course.
8. Records of training are maintained and used by management.

Level 3: Integrated Software Management

Integrated software management involves establishing and maintaining the project's defined software process and managing the software activities according to this defined software process and the special needs of the project. The project's defined software process integrates the management and technical processes as the basis for performing the project's activities. When project objectives are not being achieved, the managers know what actions need to be taken to correct the problem and reduce the likelihood of similar problems in the future. The organization provides support and historical data which the project uses to improve its software estimating, planning, and tracking process.

The goals of integrated software management are:

1. The planning and managing of each software project is based on the organization's standard software process.
2. 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.

The top-level activities performed for integrated software management are:

1. The project's defined software process is developed by tailoring the organization's standard software process according to organizational guidelines and criteria.
2. Each project's defined software process is revised according to a documented procedure.
3. The management of the software project is based on the project's defined software process.
4. The software project is staffed, trained, and managed to fulfill the special needs of the project and its defined software process.
5. The software process database for the organization is used for software planning and estimating according to a documented procedure.
6. The project's software size is quantitatively managed according to a documented procedure.
7. The project's software costs are managed against the key tasks of the defined software process according to a documented procedure.
8. The project's critical target computer resources are managed according to a documented procedure.
9. The critical dependencies and critical paths of the project's software schedule are managed according to a documented procedure.
10. The project's software risks are identified, assessed, documented, and managed according to a documented procedure.
11. A review of the software project is periodically performed to determine the actions needed to bring the project's performance and results in line with the current and projected business, customer, and end-user needs.

Level 3: Software Product Engineering

Software product engineering involves performing the technical activities to build and maintain the software system using appropriate state-of-the-practice tools and methods. The software requirements are identified, analyzed, refined, and documented. A software architecture and software designs are developed to implement the software requirements. The program code is developed to implement the software architecture and software designs. The software evaluation activities ensure that the software product to be delivered satisfies the specified requirements. A balance of flexibility and control is maintained to correct and revise the requirements, designs, and code to incorporate corrections and enhancements identified throughout the life cycle.

The goals of software product engineering are:

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

The top-level activities performed for software product engineering are:

1. Appropriate state-of-the-practice software engineering tools and methods are integrated with the project's defined software process and software life cycle.
2. The software engineering group actively participates in the control of the system requirements allocated to software.
3. The software requirements are developed, documented, and verified by systematically analyzing the allocated requirements according to the project's defined software process.
4. The software design is developed, documented, and verified, according to the project's defined software process, to accommodate the software requirements and to form the framework for coding.
5. The software code is developed and verified, according to the project's defined software process, to implement the software requirements and software design.
6. Effective techniques are used to test the software at all levels of testing.
7. Formal system testing of the software is performed, according to the project's defined software process, to ensure that the software satisfies the software requirements.
8. Acceptance testing of the software is performed, according to the project's defined software process and approved acceptance test plan, to demonstrate to the customer and end users that the software satisfies the allocated requirements.
9. The documentation that will be used to operate and maintain the system is reviewed and approved by the customer, end users, and system maintainers.
10. Data on defects identified in peer reviews and system testing of the software are collected and analyzed according to a documented procedure.
11. Consistency is maintained across software engineering products, including the software plans, allocated requirements, software requirements specification, software design, code, test plans, and test procedures.

Level 3: Intergroup Coordination

Intergroup coordination involves the disciplined interaction and coordination of the project groups with each other to address system-level issues and activities. System-level objectives and plans are established and used as the cornerstone for all project activities. The project groups participate, as appropriate, in defining the system requirements; establishing a system configuration; allocating requirements to hardware, software, firmware, and manual processes; monitoring and reviewing the hardware and software design and development; and managing and controlling changes to the system throughout the development effort. The technical working interfaces and interactions between groups are planned and managed to ensure the quality and integrity of the entire system.

The goals of intergroup coordination are:

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

The top-level activities performed for intergroup coordination are:

1. The software engineering group and the other project engineering groups actively participate with the customer and end users to establish their system needs.
2. Representatives of the software engineering group work with representatives of the other project engineering groups to monitor and coordinate the project-level technical activities and resolve project-level technical issues.
3. A documented plan is used to communicate intergroup commitments and coordinate and track the work performed.
4. Critical dependencies are identified and negotiated according to a documented procedure.
5. Products produced as input to other project groups are reviewed by representatives of the receiving groups to ensure that the products match their needs.
6. All intergroup issues affecting software are documented, negotiated, and, if unresolved, reported to the appropriate managers.
7. Periodic technical reviews and interchanges are held with the task leaders of the project groups.

Level 3: Peer Reviews

Peer reviews involve a methodical examination of work products by the producer's peers to identify defects and areas where changes and improvements are needed. The specific work products that will undergo peer review are identified as part of the software planning activities. Peer reviews are performed following defined procedures. These procedures cover preparing for the review, conducting the review, reporting the results of the review, and certifying review readiness/completion criteria. Actions identified in the review findings are documented and tracked until they are resolved.

The goals of peer reviews are:

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

The top-level activities performed for peer reviews are:

1. Peer reviews are planned and the plans are documented.
2. Peer reviews are performed according to a documented procedure.
3. Information on the conduct and results of peer reviews is recorded in an organizational database.

Level 4: Process Measurement and Analysis

Process measurement and analysis involves taking measurements of the performance of the organization's standard software process (as instantiated by the projects), analyzing these measurements, and making adjustments to stabilize the process performance within acceptable limits. The process and the associated measurements are established as a baseline and are used to plan and control the process in quantitative terms.

The goals of process measurement and analysis are:

1. The organization's standard software process is stable and under statistical process control.
2. The relationship between product quality, productivity, and product development cycle time is understood in quantitative terms.
3. 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.

The top-level activities performed for process measurement and analysis are:

1. A documented and approved plan is used as the basis for the organization's and projects' activities for process measurement and analysis.
2. The organization's standard software process is the basis for selecting the data to be collected and the analyses to be performed.
3. Process and product metrics are identified based on their usefulness to the organization and the projects.
4. Process and product data are collected according to a documented procedure.
5. The analysis of the selected process data is performed according to a documented procedure.
6. The results from the process data analyses are used to bring the organization's standard software process and its critical subprocesses under statistical process control.
7. The process performance baseline for the organization's standard software process is monitored on a regular basis and updated as appropriate.
8. The results of the measurement and analysis activities are monitored on a regular basis and appropriate adjustments are made to keep the process performance baseline in line with the expected performance.
9. Process analysis reports are prepared and distributed to the appropriate groups.

Level 4: Quality Management

Quality management involves defining software quality goals, establishing plans to achieve these goals, and monitoring and adjusting the software plans, activities, and quality goals to improve customer and end-user satisfaction. Quantitative product quality goals are established based on the needs of the organization, customer, and end users. Plans and process quality goals are established and the project's software process is specifically adjusted to achieve the product quality goals. The software activities and results are assessed against the quality objectives on a regular basis, and corrective actions are taken to bring forecasted process and product quality in line with the goals.

The goals of quality management are:

1. 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.
2. Measurable goals for process quality are established for all groups involved in the software process.
3. The software plans, design, and process are adjusted to bring forecasted process and product quality in line with the goals.
4. Process measurements are used to manage the software project quantitatively.

The top-level activities performed for quality management are:

1. The project develops strategies to satisfy the quality needs of the organization, customer, and end users.
2. A documented and approved software quality plan for the project is the basis for the project's activities for software quality management.
3. Quantitative product quality goals are defined and revised throughout the software life cycle.
4. Quantitative process quality goals are established for the software project.
5. Software product quality goals flow down to subcontractors.
6. Quantitative quality goals for software requirements are established and tracked.
7. Quantitative quality goals for software design are established and tracked.
8. Alternative software designs are considered to meet software product quality goals and software requirements.
9. Quantitative quality goals for software code are established and tracked.
10. Quantitative quality goals for formal software tests are established and tracked.
11. When quality goals are discovered to conflict (one goal cannot be achieved without compromising another goal), the software requirements, software design, software development plan, and software quality plan are revised to reflect the necessary tradeoffs.
12. The groups involved in the software process review, agree to, and work to meet the project's quality goals for its process and products.
13. Process data are monitored to identify actions needed to satisfy the process quality goals.
14. The quality of the project's products are compared against the product's quality goals on a regular basis.
15. Corrective actions are taken by the groups involved in the software process when the quality measurements indicate process or product problems.

Level 5: Defect Prevention

Defect prevention involves analyzing defects that were encountered in the past and taking action to prevent the injection of these types of defects in current and future project activities. Software activities are systematically reviewed by those who perform them to identify the defects that were encountered, to understand the root causes of the defects, and to determine the implications of the defects on future activities. Trends are analyzed to determine the kinds of defects that were encountered in the past. Defects which are likely to recur are identified, and specific actions are taken to prevent them.

The goal of defect prevention is:

1. Sources of product defects that are inherent or repeatedly occur in the software process activities are identified and eliminated.

The top-level activities performed for defect prevention are:

1. At the beginning of a software task, the members of the team performing the task meet to prepare for the activities of that task and the related defect prevention activities.
2. Each team that performs a software task conducts a causal analysis meeting shortly after the task is completed; meetings are also conducted during the task if and when the number of defects uncovered warrants the additional meetings.
3. Periodic causal analysis meetings are conducted after products are released to the customer.
4. For software tasks of long duration, periodic in-process meetings, conforming to the practices for task kickoff meetings and causal analysis meetings, are conducted.
5. Each of the teams assigned to coordinate defect prevention activities meets regularly to review and coordinate implementation of action proposals from the causal analysis meetings.
6. Revisions to the organization's standard software process and to the projects' defined software processes resulting from defect prevention actions are incorporated according to a documented procedure.
7. A database for documenting, tracking, and coordinating defect prevention actions across the organization is used according to a documented procedure.
8. The project's software engineering staff and managers receive feedback on the status and results of the organization's and project's defect prevention activities on a regular basis.

Level 5: Technology Innovation

Technology innovation involves identifying, selecting, and evaluating new technologies, and incorporating the appropriate technologies into the organization's processes. A group acts as the focus for introducing technology innovations into the organization. By maintaining an awareness of software technology innovations throughout the world and systematically evaluating and experimenting with them, the organization selects appropriate technologies to improve its productivity and product quality. Pilot efforts are performed to assess new and unproven technologies before they are introduced across the organization. With appropriate sponsorship of the organization's management, the selected technologies are incorporated into the organization's process.

The goals of technology innovation are:

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

The top-level activities performed for technology innovation are:

1. The organization develops and maintains a plan for technology innovation.
2. The organization's and projects' staff and managers are kept aware of appropriate new technologies.
3. The group focusing on technology innovation (e.g., technology support group) is involved with the organization's staff and managers in identifying areas of technology innovation.
4. The organization analyzes its standard software process to identify areas that need or could benefit from new technology.
5. A documented procedure is followed for selecting and acquiring technologies for the organization and projects.
6. Pilot efforts for improving technology are conducted, where appropriate, before technology is introduced on a broad scale.
7. Appropriate new technologies are incorporated into the organization's standard software process and the projects' defined software processes according to a documented procedure.

Level 5: Process Change Management

Process change management involves defining process improvement goals and systematically identifying, evaluating, and implementing improvements to the organization's standard software process and the projects' defined software processes on a continuous basis. Appropriate training and incentive programs are established to allow and encourage all staff and managers to participate in these process improvement activities. Improvement opportunities are identified and evaluated for potential payback for the organization. Pilot efforts are performed to assess new and unproven process changes before they are introduced across the organization.

The goals of process change management are:

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

The top-level activities performed for process change management are:

1. The group responsible for coordinating the organization's software process activities (e.g., software engineering process group) coordinates the process improvement activities.
2. The organization prepares and maintains plans that govern its activities for software process improvement.
3. The organization's software process improvement goals and accomplishments reflect the business and strategic operating plans.
4. Senior management regularly reviews the training, recognition, and motivational programs that support the activities for software process improvement and initiates changes to maintain a high level of employee participation.
5. Process improvement proposals are submitted by individuals or teams according to a documented procedure.
6. A documented procedure is followed for reviewing, approving, planning, implementing, and tracking process improvement proposals.
7. Where appropriate, the process improvements are installed on a pilot basis to determine their benefits and effectiveness before they are introduced on a broad scale.
8. Staff and managers actively participate in working groups, quality circles, or technical committees to develop process improvements for assigned process focus areas.
9. Process change recommendations resulting from completed process improvement actions are incorporated into the organization's standard software process and the projects' defined software processes according to a documented procedure.
10. A database containing process improvement information is maintained to manage the process improvement proposals.
11. The organization's staff and managers receive feedback on the status and results of the process improvement activities on a regular basis.

Abridged Practices

Appendix D: Change History

Date	Version	Change Description
1 Mar 90	0.0	Rough draft of the key practice tables; distributed to the CMM User Working Group. Version for review at the March 1990 CMM Workshop.
1 May 90	0.1	Rough draft of the key practice tables appendix to incorporate recommendations made at the March 1990 CMM Workshop against version 0.0. Version for internal SEI peer review.
6 Jun 90	0.2	Draft, revision to the key practice tables to incorporate comments from the SEI peer review of version 0.1. Version distributed to the CMM User Working Group for its review and comments. (Baselined version of the key practice tables).
26 Feb 91	0.3	Draft, revision to the key practice tables (Level 2 key process areas only) to incorporate comments from the CMM User Working Group and the Questionnaire Advisory Board made against version 0.2. Version for internal SEI peer review. Rough draft (first version) of overview and definitions to assist in the SEI peer review of the key practice tables, Level 2 key process areas.
18 Mar 91	0.4	Draft, revision to the key practice tables, Level 2 key process areas to incorporate comments from the SEI peer review of version 0.3. Version distributed to the CMM User Working Group and Questionnaire Advisory Board for their review and comments. Rough draft, revision of overview and definitions to assist the CMM User Working Group and the Questionnaire Advisory Board in their review of the key practice tables, Level 2 key process areas.

Change History

10 Apr 91	0.41	Draft, revision to the key practice tables (Level 3 key process areas only, except Software Product Engineering key process area is not included) to incorporate comments from the CMM User Working Group and the Questionnaire Advisory Board made against version 0.2. Version for internal SEI peer review.
22 Apr 91	0.5	Draft, revision to the key practice tables, Level 3 key process areas (except the Software Product Engineering key process area is not included) to incorporate comments from the SEI peer review of version 0.41. Version distributed to the CMM User Working Group and Questionnaire Advisory Board for their review and comments. Rough draft, revision of overview and definitions to assist the CMM User Working Group and the Questionnaire Advisory Board in their review of the key practice tables, Level 3 key process areas.
17 May 91	0.51	Draft, revision to the Level 2 key practice areas to incorporate comments from the CMM User Working Group made against version 0.4 and make practices of each key process area at a single maturity level. Version for internal SEI peer review.
23 May 91	0.52	Draft, revision to the key practice tables (initial version of Software Requirements Management key process area, and consolidation of Software Requirements Analysis, Software Design, and Software Testing key process areas and addition of coding practices into the single key process area, Software Product Engineering) to incorporate comments from the CMM User Working Group and the Questionnaire Advisory Board. Version for internal SEI peer review.
5 Jun 91	0.53	Preliminary baseline, revision to the Level 2 key practice areas to incorporate comments from the internal SEI peer review of version 0.51. Version submitted to SEI's Information Management.

Change History

7 Jun 91	0.54	Draft, revision to the key practice tables (initial version of Software Project Management key process area only). Version for internal SEI peer review.
14 Jun 91	0.6	Draft, revision to the key practice tables to incorporate comments from the SEI peer review of Software Requirements Management (version 0.52), Software Project Management (version 0.54), and Software Product Engineering (version 0.52) key process areas; incorporated version 0.53 of other Level 2 key process areas. Version distributed to the CMM User Working Group and Questionnaire Advisory Board for their review and comments. Rough draft, revision of overview and definitions to assist the CMM User Working Group and the Questionnaire Advisory Board in their review of the key practice tables (Software Requirements Management, Software Project Management, and Software Product Engineering key process areas). Significant changes include (1) restructuring key process area practice tables by changing the common key features, (2) modifying the set of key process areas, and (3) rearranging the practice statements so that each key process area contains practices for a single maturity level.
21 Jun 91	0.61	Draft, revision to the key practice tables (Level 4 and 5 key process areas only) to incorporate comments from the CMM User Working Group and the Questionnaire Advisory Board made against version 0.2. Version for internal SEI peer review.
28 Jun 91	0.7	Draft, revision to the key practice tables appendix, (Level 4 and 5 key process areas only) to incorporate comments from the SEI peer review of version 0.61. Version distributed to the CMM User Working Group and Questionnaire Advisory Board for their review and comments.

Change History

10 Jul 91	0.71	Draft, revision to the Level 3 key process areas and Software Requirements Management key process area to incorporate comments from the CMM User Working Group made against versions 0.5 and 0.6; make practices of each key process area at a single maturity level; and renamed three key process areas (Software Requirements Management to Requirements Management, Software Project Management to Integrated Software Management, and Technical Team Coordination to Intergroup Coordination). Version for internal SEI peer review.
14 Jul 91	0.72	Draft, revision to the Level 3 key process areas and Software Requirements Management key process area to incorporate comments from the internal SEI peer review against version 0.71. Version for internal SEI peer review.
15 Jul 91	0.73	Draft, revision to the Level 3 key process areas and Software Requirements Management key process area to incorporate comments from the internal SEI peer review.
15 Jul 91	0.74	Preliminary baseline, revision to the Level 3 key process areas and Software Requirements Management key process area (reformatting and fixing editorial defects). Version submitted to SEI's Information Management.
22 Jul 91	0.75	Draft, revision to the Level 4 and 5 key process areas to incorporate comments from the CMM User Working Group made against version 0.7. Version for internal SEI peer review.
30 Jul 91	0.76	Preliminary baseline, revision to Level 4 and 5 key process areas to incorporate comments from internal SEI peer review. Version submitted to SEI's Information Management.

Change History

30 Jul 91	0.77	Draft, revision to the preliminary baseline of the Requirements Management key process area to incorporate comments from the 30 July Questionnaire Advisory Board meeting. Version for internal SEI peer review
5 Aug 91	0.78	Draft, revision to version 0.77 of the Requirements Management key process area to incorporate comments from the 31 July Questionnaire Advisory Board meeting. Version for internal SEI peer review.
7 Aug 91	0.79	Revised preliminary baseline of the Requirements Management key process area to incorporate comments from the Questionnaire Advisory Board and the internal SEI peer review against version 0.78.
7 Aug 91	0.80	Draft, revision of overview and definitions to incorporate comments received from the CMM User Working Group and the Questionnaire Advisory Board in their review of versions 0.4, 0.5, and 0.6 and restructure the document. Version for internal SEI peer review.
15 Aug 91	1.0	Baseline version for public release.

Change History

Change Request for "Key Practices of the Capability Maturity Model"

SEI Assigned Tracking Number: _____

Product Name: Key Practices of the Capability Maturity Model

Product Version Number: 1.0

Name of Submitting Organization: _____

Organization Contact: _____ Telephone: _____

Mailing Address:

Date: _____ Short Title: _____

Change Location Tag (use section or paragraph
#, figure #, key process area ID, practice ID, etc.): _____

Proposed Change:

Rationale for Change:

Note: For the SEI to take appropriate action on the change request, we must have a clear description of the recommended change along with supporting rationale.

Send US Mail To: CMM Change Requests, Software Process Program, Software Engineering
Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890

Send Packages To: CMM Change Requests, Software Process Program, Software Engineering
Institute, Carnegie Mellon University, 4500 Fifth Avenue, Pittsburgh, PA
15213-2691

Send via Internet To: cmmchange@sei.cmu.edu
