

McCall's Software Quality Model, ISO 9126 and CMM Model

McCall Software Quality Model

- McCall Software Quality Model

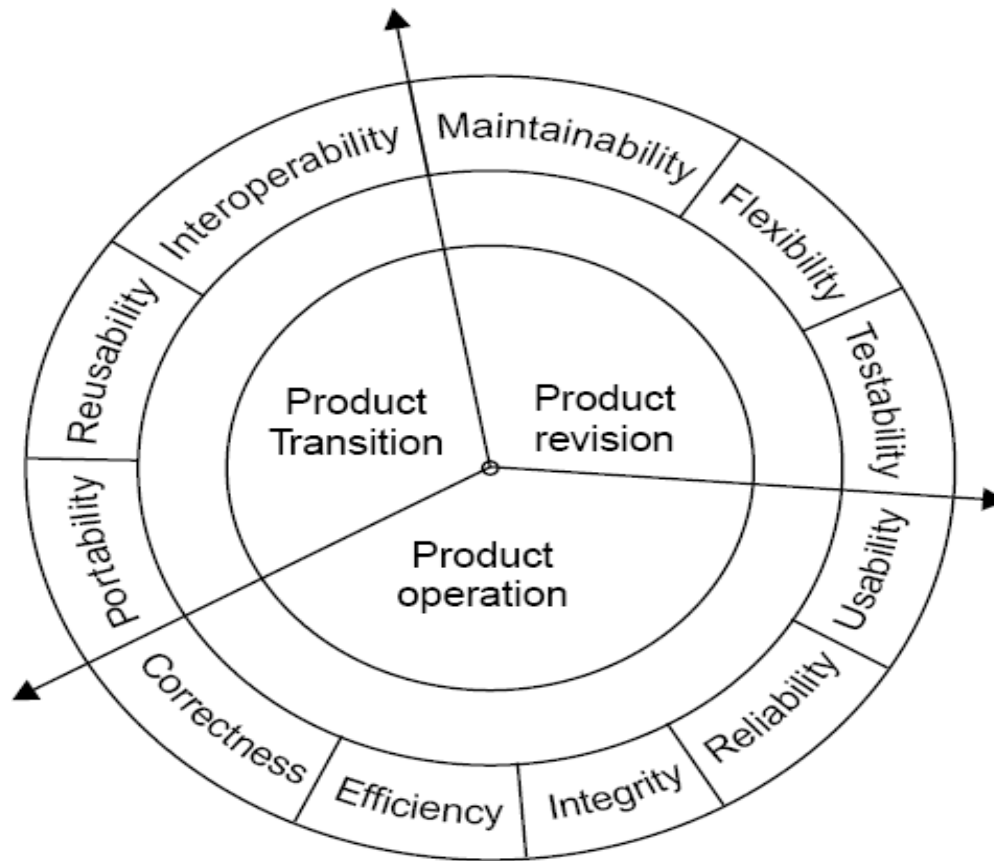


Fig 7.9: Software quality factors


McCall Software Quality Model

i. Product Operation

Factors which are related to the operation of a product are combined. The factors are:

- Correctness
- Efficiency
- Integrity
- Reliability
- Usability

These five factors are related to operational performance, convenience, ease of usage and its correctness. These factors play a very significant role in building customer's satisfaction.



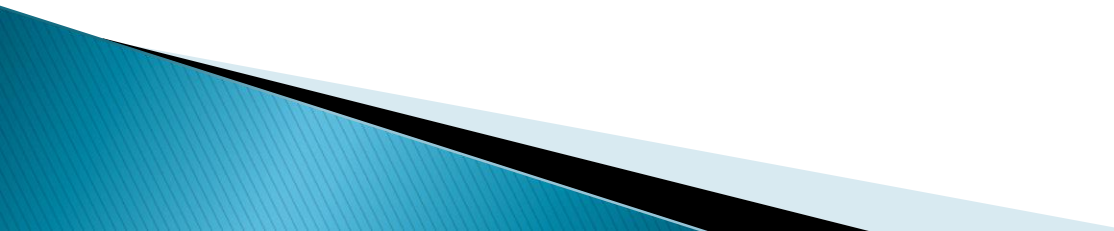
McCall Software Quality Model

ii. Product Revision

The factors which are required for testing & maintenance are combined and are given below:

- Maintainability
- Flexibility
- Testability

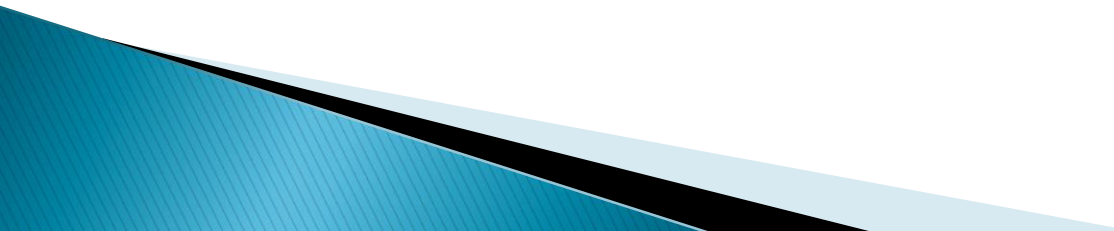
These factors pertain to the testing & maintainability of software. They give us idea about ease of maintenance, flexibility and testing effort. Hence, they are combined under the umbrella of product revision.



McCall Software Quality Model

iii. Product Transition

We may have to transfer a product from one platform to an other platform or from one technology to another technology. The factors related to such a transfer are combined and given below:

- Portability
 - Reusability
 - Interoperability
- 

Quality criteria

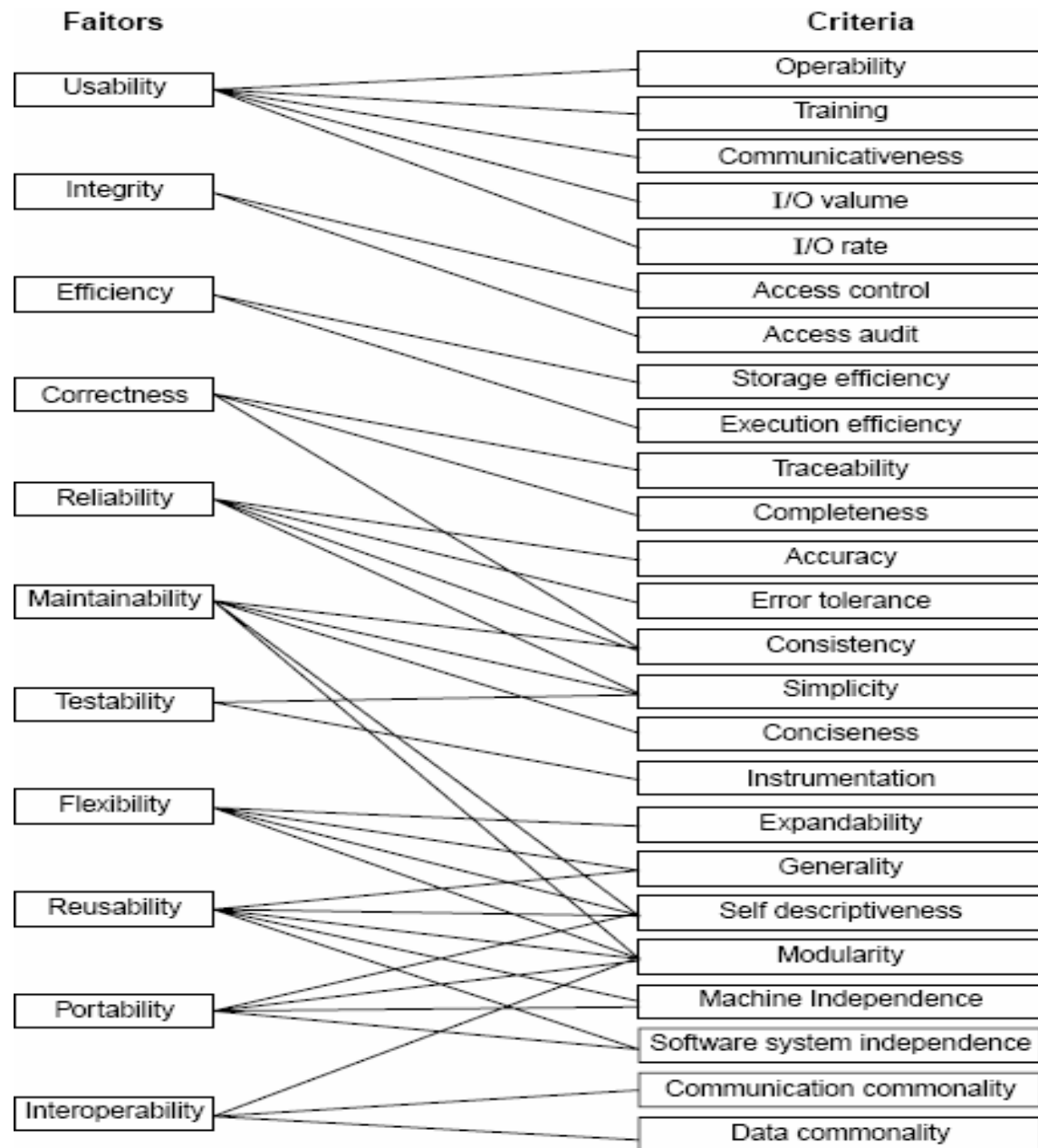


Fig 7.10: McCall's quality model

McCall Software Quality Model

<i>Sr. No.</i>	<i>Quality Criteria</i>	<i>Usability</i>	<i>Integrity</i>	<i>Efficiency</i>	<i>Correctness</i>	<i>Reliability</i>	<i>Maintainability</i>	<i>Testability</i>	<i>Flexibility</i>	<i>Reusability</i>	<i>Portability</i>	<i>Interoperability</i>
1.	Operability	x										
2.	Training	x										
3.	Communicativeness	x										
4.	I/O volume	x										
5.	I/O rate	x										
6.	Access control		x									
7.	Access Audit		x									
8.	Storage efficiency			x								
9.	Execution Efficiency			x								
10.	Traceability				x							
11.	Completeness				x							
12.	Accuracy					x						
13.	Error tolerance					x						
14.	Consistency				x	x	x					
15.	Simplicity					x	x	x				
16.	Conciseness						x					
17.	Instrumentation							x				
18.	Expandability								x			
19.	Generality								x	x		
20.	Self-descriptiveness						x		x	x	x	
21.	Modularity						x		x	x	x	x
22.	Machine independence									x	x	
23.	S/W system independence									x	x	
24.	Communication commonality											x
25.	Data commonality											x

Table 7.5(a):
Relation
between quality
factors and
quality criteria

McCall Software Quality Model

1	Operability	The ease of operation of the software.
2	Training	The ease with which new users can use the system.
3	Communicativeness	The ease with which inputs and outputs can be understood.
4	I/O volume	It is related to the I/O volume.
5	I/O rate	It is the indication of I/O rate.
6	Access control	The provisions for control and protection of the software and data.
7	Access audit	The ease with which software and data can be checked for compliance with standards or other requirements.
8	Storage efficiency	The run time storage requirements of the software.
9	Execution efficiency	The run-time efficiency of the software.

(Contd.)...

McCall Software Quality Model

10	Traceability	The ability to link software components to requirements.
11	Completeness	The degree to which a full implementation of the required functionality has been achieved.
12	Accuracy	The precision of computations and output.
13	Error tolerance	The degree to which continuity of operation is ensured under adverse conditions.
14	Consistency	The use of uniform design and implementation techniques and notations throughout a project.
15	Simplicity	The ease with which the software can be understood.
16	Conciseness	The compactness of the source code, in terms of lines of code.
17	Instrumentation	The degree to which the software provides measurements of its use or identification of errors.

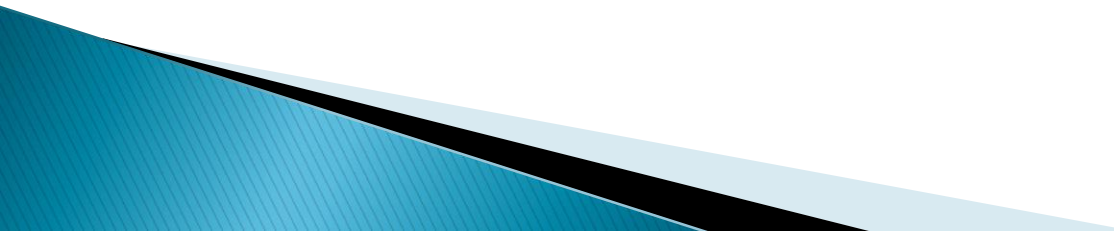
McCall Software Quality Model

18	Expandability	The degree to which storage requirements or software functions can be expanded.
19	Generability	The breadth of the potential application of software components.
20	Self-descriptiveness	The degree to which the documents are self explanatory.
21	Modularity	The provision of highly independent modules.
22	Machine independence	The degree to which software is dependent on its associated hardware.
23	Software system independence	The degree to which software is independent of its environment.
24	Communication commonality	The degree to which standard protocols and interfaces are used.
25	Data commonality	The use of standard data representations.

Table 7.5 (b): Software quality criteria

ISO 9126

Was standardized in 1992. Provides a single model to standardise the quality factors. Identifies following six factors for quality

- Functionality
 - Reliability
 - Usability
 - Efficiency
 - Maintainability
 - Portability
- 

ISO 9126

Characteristic / Attribute	Short Description of the Characteristics and the concerns Addressed by Attributes
Functionality	Characteristics relating to achievement of the basic purpose for which the software is being engineered
• Suitability	The presence and appropriateness of a set of functions for specified tasks
• Accuracy	The provision of right or agreed results or effects
• Interoperability	Software's ability to interact with specified systems
• Security	Ability to prevent unauthorized access, whether accidental or deliberate, to program and data.
Reliability	Characteristics relating to capability of software to maintain its level of performance under stated conditions for a stated period of time
• Maturity	Attributes of software that bear on the frequency of failure by faults in the software

(Contd.)...

ISO 9126

• Fault tolerance	Ability to maintain a specified level of performance in cases of software faults or unexpected inputs
• Recoverability	Capability and effort needed to re-establish level of performance and recover affected data after possible failure.
Usability	Characteristics relating to the effort needed for use, and on the individual assessment of such use, by a stated implied set of users.
• Understandability	The effort required for a user to recognize the logical concept and its applicability.
• Learnability	The effort required for a user to learn its application, operation, input and output.
• Operability	The ease of operation and control by users.
Efficiency	Characteristic related to the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

(Contd.)...

ISO 9126

• Time behavior	The speed of response and processing times and throughout rates in performing its function.
• Resource behavior	The amount of resources used and the duration of such use in performing its function.
Maintainability	Characteristics related to the effort needed to make modifications, including corrections, improvements or adaptation of software to changes in environment, requirements and functions specifications.
• Analyzability	The effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.
• Changeability	The effort needed for modification, fault removal or for environmental change.
• Stability	The risk of unexpected effect of modifications.
• Testability	The effort needed for validating the modified software.

(Contd.)...

ISO 9126

Portability	Characteristics related to the ability to transfer the software from one organization or hardware or software environment to another.
• Adaptability	The opportunity for its adaptation to different specified environments.
• Installability	The effort needed to install the software in a specified environment.
• Conformance	The extent to which it adheres to standards or conventions relating to portability.
• Replaceability	The opportunity and effort of using it in the place of other software in a particular environment.

Table 7.6: Software quality characteristics and attributes – The ISO 9126 view

ISO 9126

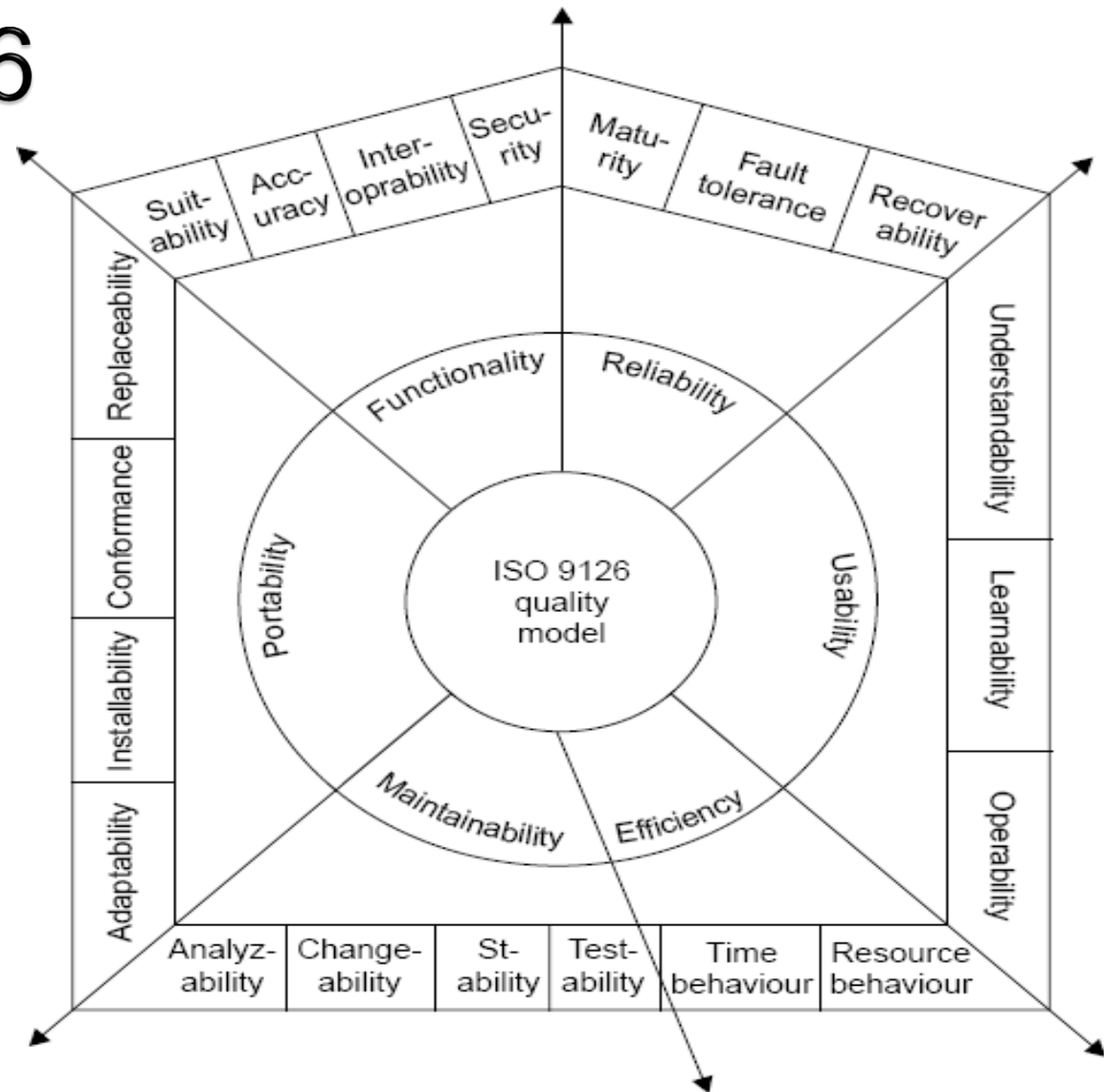


Fig.7.12: ISO 9126 quality model

Capability Maturity Model

- Capability Maturity Model

It is a strategy for improving the software process, irrespective of the actual life cycle model used.

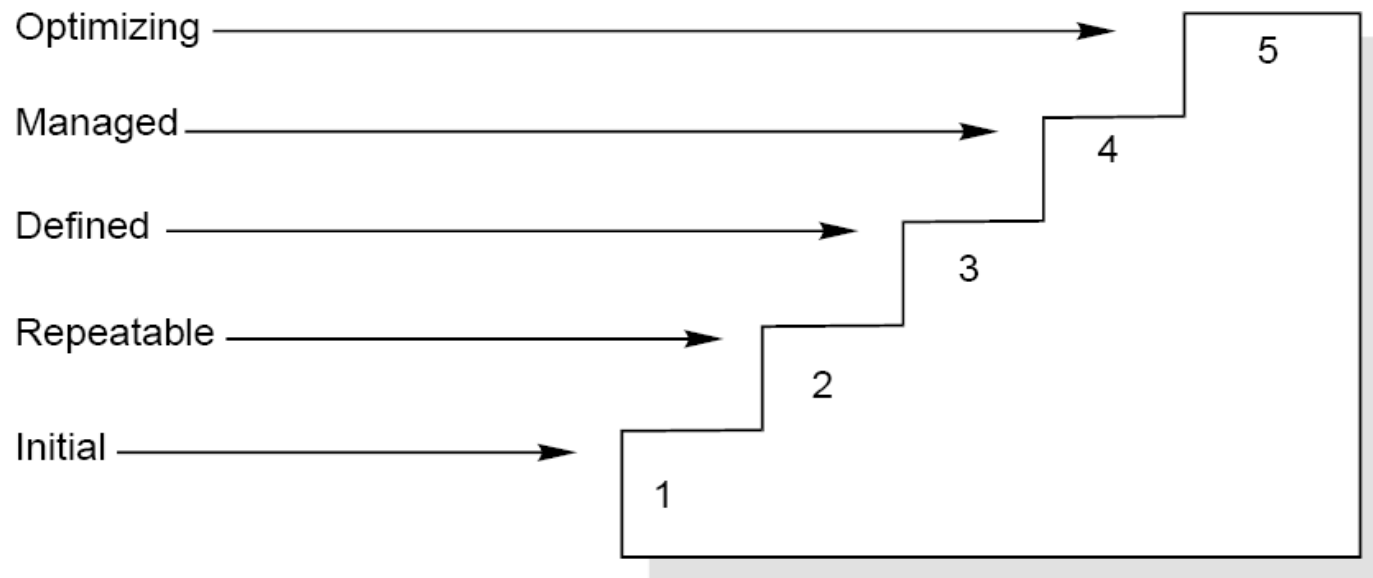
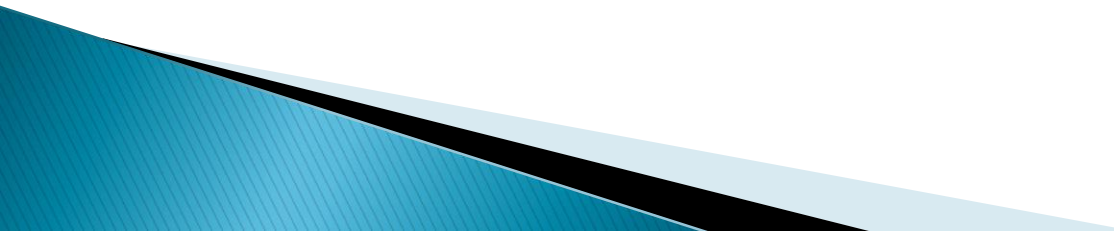


Fig.7.23: Maturity levels of CMM

Capability Maturity Model

Maturity Levels:

- ✓ Initial (Maturity Level 1)
 - ✓ Repeatable (Maturity Level 2)
 - ✓ Defined (Maturity Level 3)
 - ✓ Managed (Maturity Level 4)
 - ✓ Optimizing (Maturity Level 5)
- 

Capability Maturity Model

Maternity Level	Characterization
Initial	Adhoc Process
Repeatable	Basic Project Management
Defined	Process Definition
Managed	Process Measurement
Optimizing	Process Control

Fig.7.24: The five levels of CMM

Capability Maturity Model

- Key Process Areas

The key process areas at level 2 focus on the software project's concerns related to establishing basic project management controls, as summarized below:

Requirements Management (RM)

Establish a common relationship between the customer requirements and the developers in order to understand the requirements of the project.

Software Project Planning (PP)

Establish reasonable plans for performing the software engineering and for managing the software project.

Software Project Tracking and Oversight (PT)

Establish adequate visibility into actual progress so that management can take effective actions when the software project's performance deviates significantly from the software plans.

Software Subcontract Management (SM)

Select qualified software subcontractors and manage them effectively.

Software Quality Assurance (QA)

Provide management with appropriate visibility into the process being used by the software project and of the products being built.

Software Configuration Management (CM)

Establish and maintain the integrity of the products of the software project throughout the project's software life cycle.

Capability Maturity Model

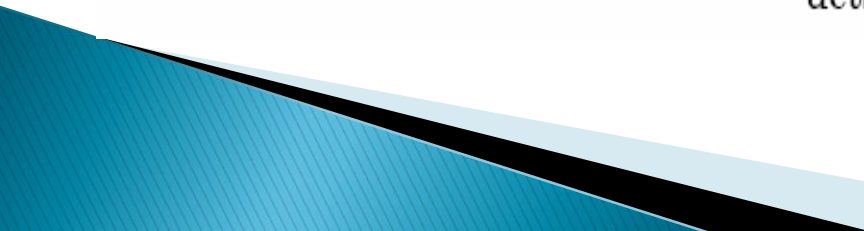
The key process areas at level 3 address both project and organizational issues, as summarized below:

Organization Process Focus (PF)	Establish the organizational responsibility for software process activities that improve the organization's overall software process capability.
Organization Process Definition (PD)	Develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for cumulative, long-term benefits to the organization.
Training Program (TP)	Develop the skills and knowledge of individuals so that they can perform their roles effectively and efficiently.
Integrated Software Management (IM)	Integrate the software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets.

(Contd.)...

Capability Maturity Model

Software Product Engineering (PE)	Consistently perform a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently.
Inter group Coordination (IC)	Establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently.
Peer Reviews (PR)	Remove defects from the software work products early and efficiently. An important corollary effect is to develop a better understanding of the software work products and of the defects that can be prevented.



Capability Maturity Model

The key process areas at level 4 focus on establishing a quantitative understanding of both the software process and the software work products being built, as summarized below:

Quantitative Process
Management (QP)

Control the process performance of the software project quantitatively.

Software Quality Management (QM)

Develop a quantitative understanding of the quality of the project's software products and achieve specific quality goals.

Capability Maturity Model

The key process areas at level 5 cover the issues that both the organization and the projects must address to implement continuous and measurable software process improvement, as summarized below:

Defect Prevention (DP)	Identify the causes of defects and prevent them from recurring.
Technology Change Management (TM)	Identify beneficial new technologies (i.e., tools, methods, and processes) and transfer them into the organization in an orderly manner.
Process Change Management (PC)	Continually improve the software processes used in the organization with the intent of improving software quality, increasing productivity, and decreasing the cycle time for product development.

