6) Software Quality Management

Basic Quality Concepts:-

Quality refers to the features and characteristics of product which define its ability to satisfy user requirements.

Various Quality concepts are:

* Ease of use and operation
* Confirmity with established standards and end user requirements.
* Well Documented
* Performance as expected by user
* Ease of modification
* Well tested for functionality, user interface, performance, etc.

Quality:- It is a measureable characteristic of product. It means satisfying customer needs regarding requirements, cost, service, delivery and schedule.

IEEE Def:- The degree to which the system, components or process meets specified requirements.

Software Quality is good if:-

* It has fewer failures
* It is reliable
* It satisfies majority of customer

Types of quality:

① Quality of Design:- Refers to characteristics and design specified for the end product to be constructed. It includes requirements, specification and the design of the system.

(2) Quality of Confirmance :- Degree to which design specifications are followed in manufacturing the product. It primarily focuses on implementation.

Quality Control :- It encompasses a set of software engineering actions that help to ensure that each product meets its quality goals.

Quality Control Activities :-

* Models are reviewed to ensure that they are complete and consistent.
* Code is inspected to uncover and correct errors before testing commences.
* A series of testing steps is applied to uncover errors in processing logic, data manipulation & interface communication.
* Defining and classifying severity of objects.
* Recording defects
* Tracking corrective action on defects and defect data analysis.
* A combination of measurement and feedback allows a software team to tune the process when any of these work products fail to meet quality goals.

Software Quality Assurance (SQA):-

Software Quality assurance (Quality management) is an umbrella activity that is applied throughout the software process.

SQA encompasses:-

* An SQA process
* Specific quality assurance & quality control tasks
* Effective software engineering practice
* Control of all software work product and changes made to them
* A procedure to ensure compliance with software development standards.
* Measurement and reporting mechanisms.

Activities/Elements of SQA:-

① Standards:- The job of SQA is to ensure that adopted standards are followed and that all work products conform them.

② Reviews & audits:- Technical reviews are a quality control activity performed by software engineers for software engineers. Their intent is to uncover errors. Audits are a type of review performed by SQA personnel with the intent of ensuring that quality guidelines are being followed for software engineering work.

③ Testing:- Software testing is a quality control function that has one primary goal - to find errors. The job of SQA is to ensure that testing is properly planned and efficiently conducted so that it has the highest chances of finding errors.

④ Error/defect collection analysis :- The SQA collects and analyses error and defect data to better understand how errors are introduced and what SE activities are best suited to remove them.

⑤ Change management :- SQA ensures that adequate change management practices are followed.

⑥ Education :- Every software organisation wants to improve its software engineering practices. A key contributor is education of software engineers, their managers and stakeholders. The SQA organisation takes lead in software process improvement and sponsor of education programs.

⑦ Vendor management :- The job of the SQA organisation is to ensure high-quality software results by suggesting specific practices that the vendor should follow and incorporating quality mandates as part of any contract with external vendor.

⑧ Security management :- SQA ensures that appropriate process and technology are used to achieve software security.

⑨ Safety :- Because software is a pivotal component of human rated system, the impact of hidden defects can be catastrophic. SQA may be responsible for assessing the impact of software failure and for initiating those steps required to reduce risk.

⑩ Risk management :- The SQA organisation ensures that risk management activities are properly conducted and the risk-related contingency plans have been established.

Statistical Software Quality Assurance:-

Statistical Software Quality Assurance reflects a growing trend throughout industry to become more quantitive about quality.

Statistical SQA steps :-

1. Information about software errors and defects is collected and organised.
2. To trace each error and defect to its underlying cause.
3. Using the Pareto principle (80% of the defects can be traced to 20% of all possible causes) isolate the 20% (the vital few).
4. Once vital few causes have been identified, move to correct the problems that have caused errors and defects.

Ex:- For one year, software engineering organisation collects information of errors :-

All problems which are uncovered can be tracked to one or more of the following causes.

* Incomple or erraneous specifications (IES)
* Misinterpretation of customer communication (MCC)
* Intentional deviations from specification (IDS)
* Violation of programming standards (VPS)
* Error in data representation (EDR)
* Inconsistent component Interface (ICI)
* Error in design logic (EDL)
* Incomplete or erraneous testing (IET)
* Inaccurate or incomplete documentation (IID)
* Error in programming language translation of design (PLT)
* Ambiguous or inconsistent human/comp. interface (HCI)
* Miscellaneous (MIS)

Then table is built:

Error	Total		Serious		Moderate		Minor	
	No.	%	No.	%	No.	%	No.	%
IES	205	22%	34	27%	68	18%	103	24%
MCC	156	17%	12	9%	68	18%	76	17%
IDS	48	5%	1	1%	24	6%	23	5%
VPS	25	3%	0	0%	15	4%	10	2%
EDR	130	14%	26	20%	68	18%	36	8%
ICI	58	6%	9	7%	18	5%	31	7%
EDL	45	5%	14	11%	12	3%	19	4%
IET	95	10%	12	9%	35	9%	48	11%
IID	36	4%	2	2%	20	5%	14	3%
PLT	60	6%	15	12%	19	5%	26	6%
HCI	28	3%	3	2%	17	4%	8	2%
MIS	56	6%	0	0%	15	4%	41	9%
Totals	942	100%	128	100%	379	100%	435	100%

The above table indicates that IES, MCC and EDR are the vital few causes that account for 53% of all errors.

It should be noted, however, that IES, EDR, PLT & EDL would be selected as the vital few causes if only serious errors are considered.

Once vital few causes are determined, the software engineering organisation can begin corrective action.

Quality Evaluation Standards:

Six sigma for software:- The term six sigma is derived from six standard deviations - 3.4 instances (defects) per million occurances - implying a extremely high quality standard.

[DMAIC] → The six sigma methodology defines three core steps:

1. Define customer requirements and deliverables and project goals via well defined methods of customer communication.
2. Measure the existing process and its output to determine current quality performance (Collect defect metrics).
3. Analyse defect metrics and determine the vital few causes.

If an existing software process is in place, but improvement is required, six sigma suggest two additional steps:

4. Improve the process by eliminating the root causes of defects.
5. Control the process to ensure that future work does not reintroduce the the causes of defects.

[DMADV] → If an organisation is developing a software process (rather than improving an existing process), the core steps are argumented as follows:

1. Design the process to (1) avoid the root causes of defects and (2) to meet customer requirements.
2. Verify that the process model will, in fact, avoid defects and meet customer requirements

ISO 9000 Standard for software:

ISO 9000 describes quality assurance elements in generic terms that can be applied to any business regardless of products and services offered. To become registered to one of the quality assurance system models contained in ISO 9000, a company's quality system and operations are scrutinized by third party auditors for compliance to the standard and for effective operation. Upon sucessful registration, a company is issued a certificate from the registration body represented by the auditors

Capability Maturity Model Integration (CMMI):-

The Capability Maturity Model Integration (CMMI) is a comprehensive process meta-model that is predicted on a set of system and software and software engineering capabilities that should be present as organisations reach different levels of process capability & maturity.

CMMI represents meta-model in two different ways:-

* As a "continuous" model
* As a "staged" model

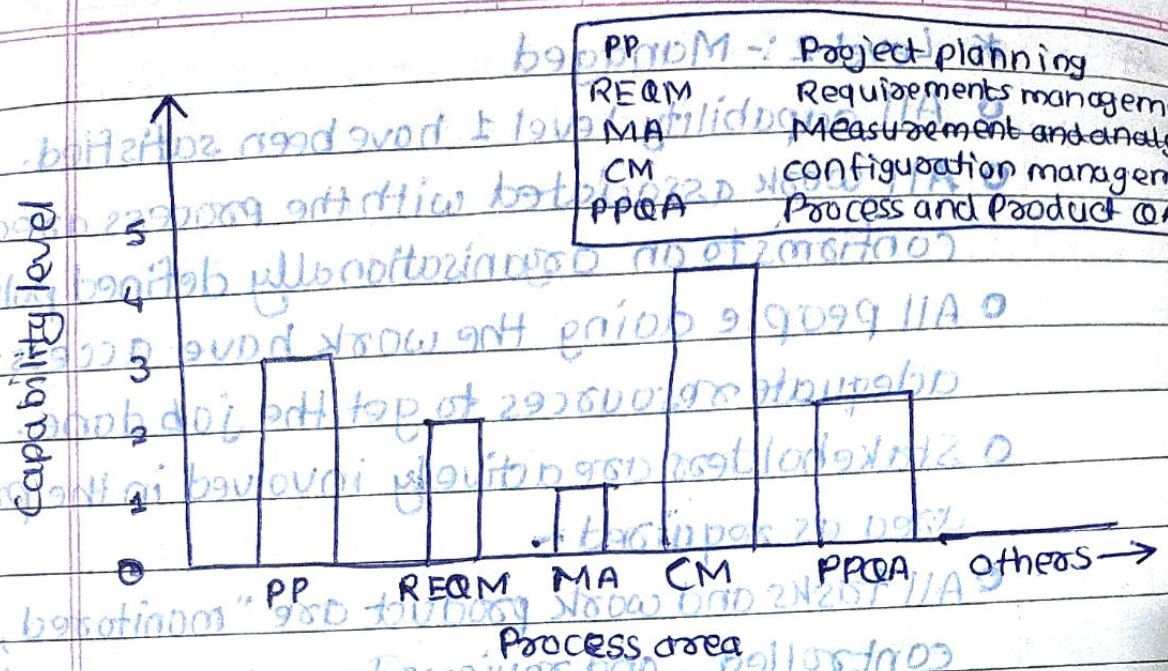


Fig :- CMMI process compatability profile

Each process area is assesed against specific goals and practices and is rated according to the following capability levels.

* Level 0 :- Incomplete
  - The process area is either not performed or does not achieve all goals and objectives defined by the CMMI for level 1 capability for process area.

* Level 1 :- Performed
  - All specific goals of the process area have been satisfied.
  - Work tasks required to produce defined work products are being conducted.

* Level 2 :- Managed
- All capability level 1 have been satisfied.
- All work associated with the process area confirms to an organisationally defined policy.
- All people doing the work have access to adequate resources to get the job done.
- Stakeholders are actively involved in the process area as required.
- All tasks and work product are "monitored, controlled, and reviewed; and evaluated".

* Level 3 :- Defined
- All capability level 2 criteria have been achieved
- The process is "tailored from organisation's set of standard processes according to the organisation's tailoring guidelines, and contributes work products, measures, and other process-improvement information to the organisational process assets."

* Quantatively ~~Mang~~ Managed (Level 4)
- All capability level have been achieved.
- The process area is controlled and improved using measurement and quantative assesment

* Level 5 :- Optimize
- All capability level 4 criteria have been achieved
- The process area is adapted and optimized using quantative means to meet changing customer needs and to continually improve the efficacy of the process area under consideration.

The CMMI defines each process area in terms of "specific goals" and the "specific practices" required to achieve these goals.

- Specific goals establish the characteristics that must exist if the activities implied by a process area are to be effective.
- Specific practices refine a goal into a set of process-related activities.

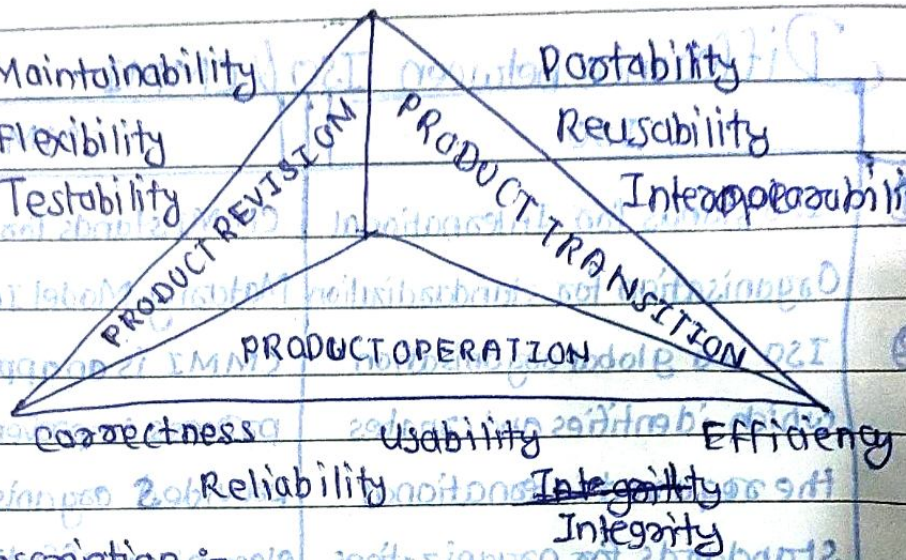
Difference between ISO & CMMI:

	ISO	CMMI
①	ISO stands for International Organisation for standardization	CMMI stands for Capability Maturity Model Integration
②	ISO is a global organisation which identifies and creates the required international standards for organizations, government bodies.	CMMI is an approach for process improvement which provides organisations, essential elements of effective processes to improve their performance.
③	ISO is referred to as an audit standard	CMMI is referred to as a process model
④	ISO is a certification tool and one organisation can get this certification after conforming some standards	Different organisations can get rating from Level 1 to Level 5 depending upon maturity processes defined in every process level.
⑤	ISO is flexible and applicable to all manufacturing industries	CMMI is rigid and extends only to business developing software intensive systems.
⑥	It addresses corporate business	It primarily focuses on software engineering activities.

McCall's Quality Factors :-

It gives useful categorisation of factors that affect software quality.
These software quality factors focus on three important aspects of a software product :-

- Its operational characteristics.
- Its ability to undergo change.
- Its adaptability to new environment.



Description :-

(1) Correctness :- The extent to which a program satisfies its specification and fulfills customer's objectives.

(2) Reliability :- The extent to which a program can be expected to perform its intended function with required precision.

(3) Usability :- Effort required for learning, operating preparing input for, and interpreting output for a program.

(4) Integrity :- Extent to which access to software or data by unauthorised persons can be controlled

(5) Efficiency :- The amount of computing resources and code required by a program to perform its function.

Date

(6) Maintainability :- Effort required for locating and fixing an error in a program.

(7) Flexibility :- Effort required for modifying an operational program.

(8) Testability :- Effort required for testing a program to ensure that it performs its intended function.

(9) Portability :- Effort required to transfer the program from one hardware and / or software system environment to another.

(10) Reusability :- Extent to which a program can be reused in other applications.

(11) Interoperability :- Effort required to couple one system to another.

It is difficult to develop direct measures of these quality factors. However accessing quality of an application using these factors will provide us with a solid indication of software quality.