Software Engineering-BSCE-301L

Module 7:

Quality Assurance

Dr . Saurabh Agrawal

Faculty Id: 20165

School of Computer Science and Engineering

VIT, Vellore-632014

Tamil Nadu, India

Outline

- Product and Process Metrics (23.1 and 25.1)
- Quality Standards Models: ISO, TQM, Six-Sigma (16.7, 16.5.2)
- Process improvement Models: CMM & CMMI (30.3 and 30.4)
- Quality Control and Quality Assurance (14.4.3 and 14.4.4)
- Quality Management (14.1)
- Quality Factors (14.2.2, 14.2.3, 14.2.4)
- Methods of Quality Management (14.4)

Product Metrics

- Product metrics are software product measures at any stage of their development, from requirements to established systems.
- Product metrics are related to software features only.
- **Product metrics fall into two classes:**

1. Dynamic metrics that are collected by measurements made from a program in execution.
2. Static metrics that are collected by measurements made from system representations such as design, programs, or documentation.

Product Metrics

- Dynamic metrics help in assessing the efficiency and reliability of a program while static metrics help in understanding, understanding and maintaining the complexity of a software system.
- Dynamic metrics are usually quite closely related to software quality attributes.
- It is relatively easy to measure the execution time required for particular tasks and to estimate the time required to start the system.
- These are directly related to the efficiency of the system failures and the type of failure can be logged and directly related to the reliability of the software.
- On the other hand, static matrices have an indirect relationship with quality attributes.
- A large number of these matrices have been proposed to try to derive and validate the relationship between the complexity, understandability, and maintainability.
- Several static metrics which have been used for assessing quality attributes, given in table of these, program or component length and control complexity seem to be the most reliable predictors of understandability, system complexity, and maintainability.

Product Metrics

❑Software Product Metrics :

S.No.	Software Metric	Description
(1)	Fan-in/Fan-out	Fan-in is a measure of the number of functions that call some other function (say X). Fan-out is the number of functions which are called by function X. A high value for fan-in means that X is tightly coupled to the rest of the design and changes to X will have extensive knock-on effects. A high value for fan-out suggests that the overall complexity of the control logic needed to coordinate the called components.
(2)	Length of code	This is measure of the size of a program. Generally, the large the size of the code of a program component, the more complex and error-prone that component is likely to be.
(3)	Cyclomatic complexity	This is a measure of the control complexity of a program. This control complexity may be related to program understandability.

Product Metrics

❑Software Product Metrics :

(4)	Length of identifiers	This is a measure of the average length of distinct identifier in a program. The longer the identifiers, the more understandable the program.
(5)	Depth of conditional nesting	This is a measure of the depth of nesting of if statements in a program. Deeply nested if statements are hard to understand and are potentially error-prone.
(6)	Fog index	This is a measure of the average length of words and sentences in documents. The higher the value for the Fog index, the more difficult the document may be to understand.

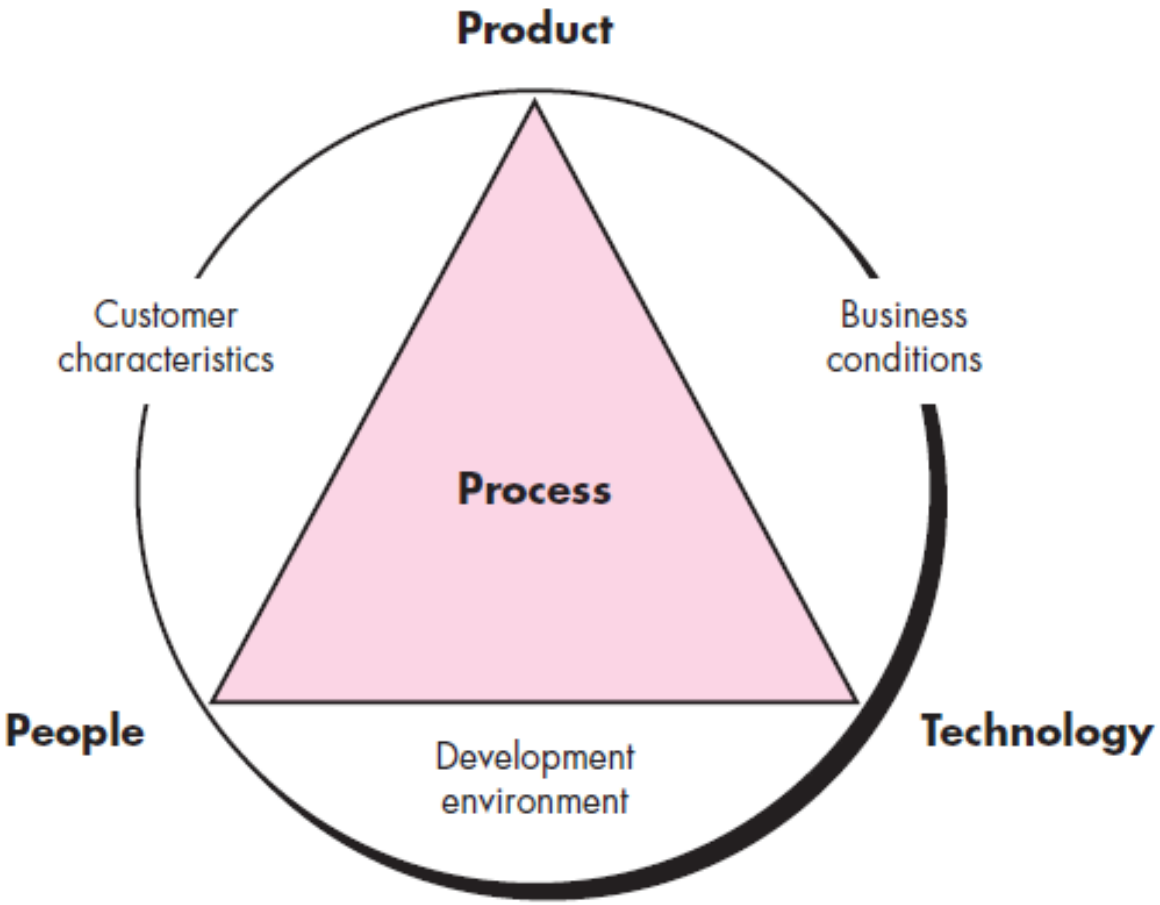
Process Metrics

- The only rational way to improve any process is to measure specific attributes of the process, develop a set of meaningful metrics based on these attributes, and then use the metrics to provide indicators that will lead to a strategy for improvement.
- But before I discuss software metrics and their impact on software process improvement, it is important to note that process is only one of a number of “controllable factors in improving software quality and organizational performance” [Pau94].
- Referring to Figure 25.1, process sits at the center of a triangle connecting three factors that have a profound influence on software quality and organizational performance.
- The skill and motivation of people has been shown [Boe81] to be the single most influential factor in quality and performance.
- The complexity of the product can have a substantial impact on quality and team performance.
- The technology (i.e., the software engineering methods and tools) that populates the process also has an impact.

Process Metrics

FIGURE 25.1

Determinants for software quality and organizational effectiveness.
Source: Adapted from [Pau94].



Process Metrics

- In addition, the process triangle exists within a circle of environmental conditions that include the development environment (e.g., integrated software tools), business conditions (e.g., deadlines, business rules), and customer characteristics (e.g., ease of communication and collaboration).
- You can only measure the efficacy of a software process indirectly.
- That is, you derive a set of metrics based on the outcomes that can be derived from the process.
- Outcomes include measures of errors uncovered before release of the software, defects delivered to and reported by end users, work products delivered (productivity), human effort expended, calendar time expended, schedule conformance, and other measures.
- You can also derive process metrics by measuring the characteristics of specific software engineering tasks.
- For example, you might measure the effort and time spent performing the umbrella activities and the generic software engineering activities.

Process Metrics

❑Software process metrics can provide significant benefit as an organization works to improve its overall level of process maturity.

❑However, like all metrics, these can be misused, creating more problems than they solve.

❑Grady [Gra92] suggests a “software metrics etiquette” that is appropriate for both managers and practitioners as they institute a process metrics program:

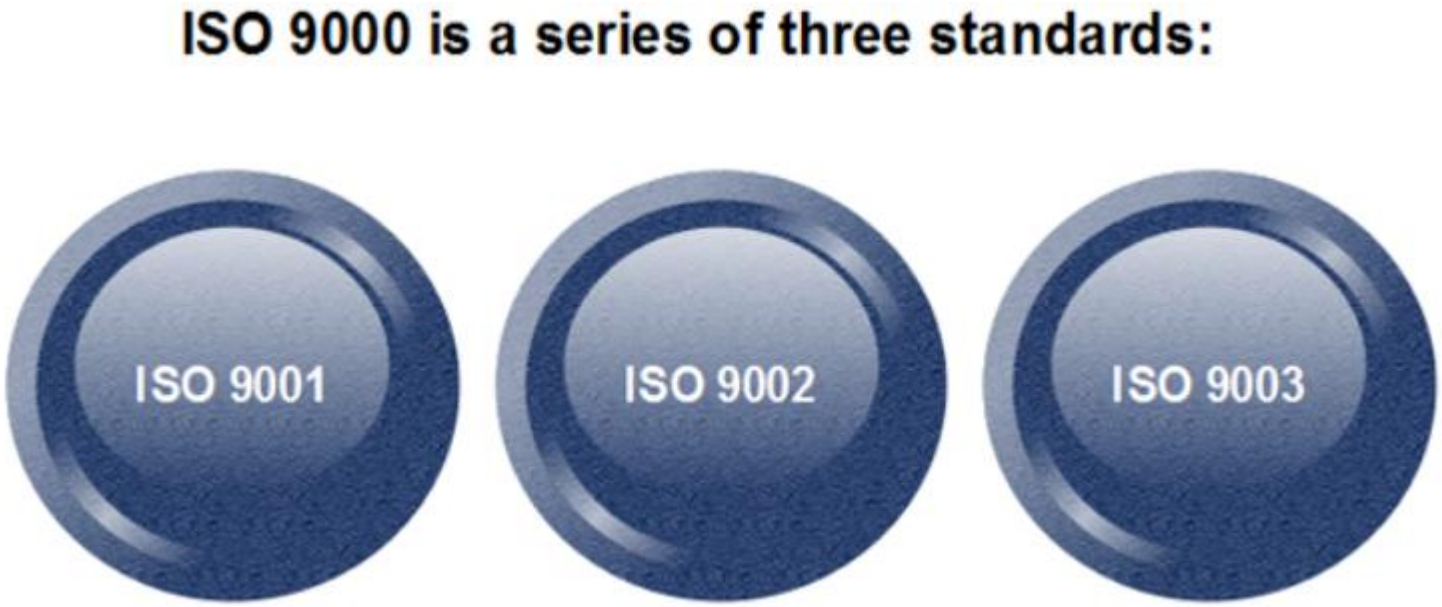
1. Use common sense and organizational sensitivity when interpreting metrics data.
2. Provide regular feedback to the individuals and teams who collect measures and metrics.
3. Don’t use metrics to appraise individuals.
4. Work with practitioners and teams to set clear goals and metrics that will be used to achieve them.
5. Never use metrics to threaten individuals or teams.
6. Metrics data that indicate a problem area should not be considered “negative.” These data are merely an indicator for process improvement.
7. Don’t obsess on a single metric to the exclusion of other important metrics.

Quality Standard Models: ISO

- ISO **(International Standards Organization)** is a group or consortium of 63 countries established to plan and fosters standardization.
- ISO declared its 9000 series of standards in 1987.
- It serves as a reference for the contract between independent parties.
- The ISO 9000 standard determines the guidelines for maintaining a quality system.
- The ISO standard mainly addresses operational methods and organizational methods such as responsibilities, reporting, etc.
- ISO 9000 defines a set of guidelines for the production process and is not directly concerned about the product itself.

Quality Standard Models: ISO

❑Types of ISO 9000 Quality Standards



Quality Standard Models: ISO

- The ISO 9000 series of standards is based on the assumption that if a proper stage is followed for production, then good quality products are bound to follow automatically.
- The types of industries to which the various ISO standards apply are as follows.
- **ISO 9001:** This standard applies to the organizations engaged in design, development, production, and servicing of goods. This is the standard that applies to most software development organizations.
- **ISO 9002:** This standard applies to those organizations which do not design products but are only involved in the production. Examples of these category industries contain steel and car manufacturing industries that buy the product and plants designs from external sources and are engaged in only manufacturing those products. Therefore, ISO 9002 does not apply to software development organizations.
- **ISO 9003:** This standard applies to organizations that are involved only in the installation and testing of the products. For example, Gas companies.

Quality Standard Models: ISO

- How to get ISO 9000 Certification: An organization determines to obtain ISO 9000 certification applies to ISO registrar office for registration.
- The process consists of the following stages:

Quality Standard Models: ISO

❑How to get ISO 9000 Certification:

- **Application:** Once an organization decided to go for ISO certification, it applies to the registrar for registration.
- **Pre-Assessment:** During this stage, the registrar makes a rough assessment of the organization.
- **Document review and Adequacy of Audit:** During this stage, the registrar reviews the document submitted by the organization and suggest an improvement.
- **Compliance Audit:** During this stage, the registrar checks whether the organization has compiled the suggestion made by it during the review or not.
- **Registration:** The Registrar awards the ISO certification after the successful completion of all the phases.
- **Continued Inspection:** The registrar continued to monitor the organization time by time.

Quality Standard Models: TQM

- Total Quality Management (TQM) is a management philosophy that fosters a culture of excellence, emphasizing continual improvement, customer satisfaction, and active employee involvement.
- TQM places a strong emphasis on understanding and meeting the needs and expectations of customers through effective communication and feedback mechanisms.
- By integrating quality principles into the very fabric of an organisation, TQM seeks to optimize processes, enhance products and services, and drive overall customer satisfaction.
- A key component of TQM is the unwavering commitment to ongoing improvement, where all employees are encouraged to identify areas for enhancement, eliminate waste, and work collaboratively toward achieving incremental gains.
- Moreover, TQM recognizes the invaluable contribution of employees and empowers them to actively participate, make decisions, and take ownership of quality initiatives.

Quality Standard Models: TQM

Quality Standard Models: TQM

❑**Principles of Total Quality Management (TQM):** The principles of Total Quality Management (TQM) encompass the following:

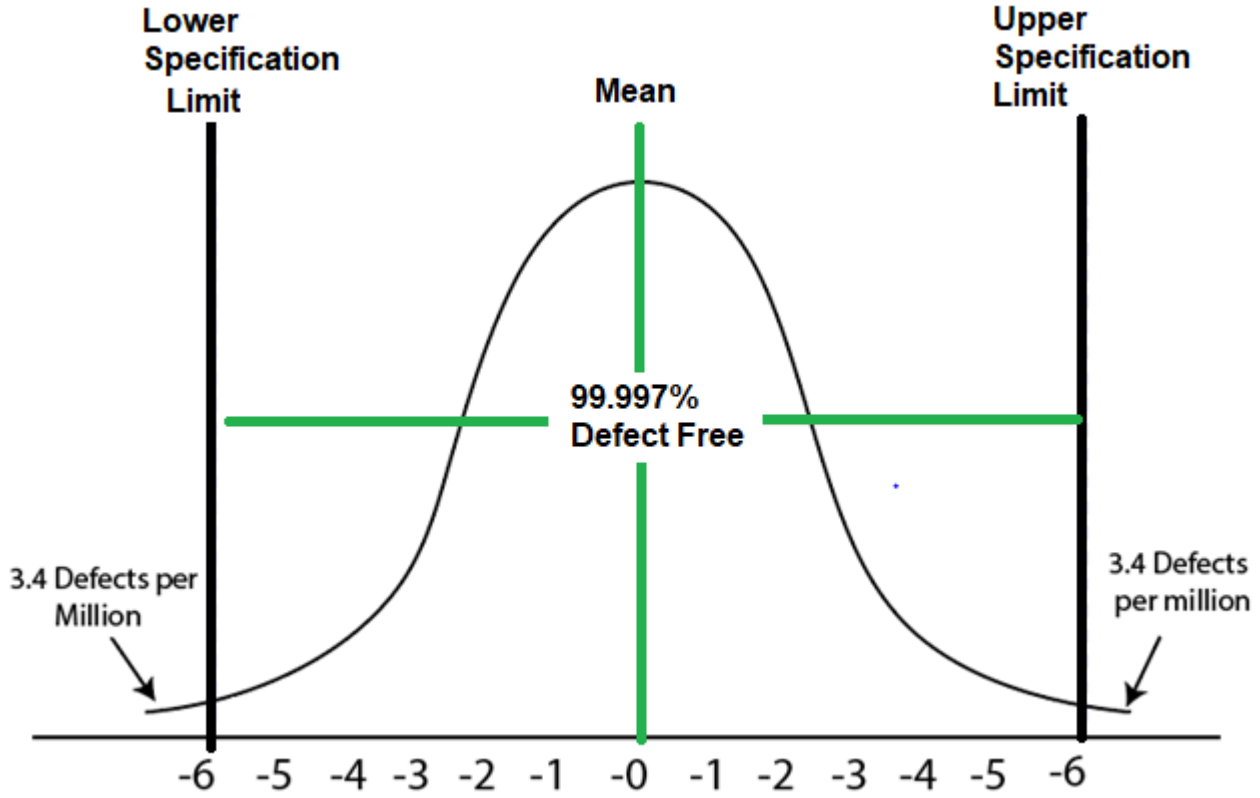
1. **Customer Focus:** TQM places a strong emphasis on understanding and meeting customer needs and expectations. It involves gathering customer feedback, conducting market research, and using that information to improve products, services, and overall customer satisfaction.
2. **Continuous Improvement:** TQM promotes a culture of continual improvement throughout the organisation. It encourages all employees to actively participate in identifying opportunities for enhancement, eliminating waste, and implementing incremental improvements in processes, products, and services.
3. **Employee Involvement:** TQM recognizes the importance of involving employees at all levels in quality improvement initiatives. It fosters a collaborative and empowered work environment, where employees are encouraged to contribute ideas, make decisions, and take ownership of quality-related activities.

Quality Standard Models: TQM

4. **Data-Driven Decision-Making:** TQM relies on the collection and analysis of relevant data to support decision-making. It emphasizes the use of facts and figures to identify areas for improvement, measure performance, and monitor progress toward quality objectives.
5. **Supplier Relationships:** TQM recognizes the significance of strong relationships with suppliers. It emphasizes collaboration, communication, and mutually beneficial partnerships with suppliers to ensure the quality of inputs and optimize the overall value chain.
6. **Leadership Commitment:** TQM requires committed leadership that actively supports and promotes quality principles throughout the organisation. Leaders serve as role models, set clear quality goals, provide necessary resources, and foster a culture that prioritizes continuous improvement and customer satisfaction.

Quality Standard Models: Six-Sigma

- Six Sigma is the process of producing high and improved quality output.
- This can be done in two phases – identification and elimination.
- The cause of defects is identified and appropriate elimination is done which reduces variation in whole processes.
- A six sigma method is one in which 99.99966% of all the products to be produced have the same features and are of free from defects.



Quality Standard Models: Six-Sigma

- **Characteristics of Six Sigma:** The Characteristics of Six Sigma are as follows:

1. **Statistical Quality Control:**

   Six Sigma is derived from the Greek Letter ? which denote Standard Deviation in statistics.

   Standard Deviation is used for measuring the quality of output.

2. **Methodical Approach:**

   The Six Sigma is a systematic approach of application in DMAIC and DMADV which can be used to improve the quality of production. DMAIC means for Design-Measure- Analyze-Improve-Control. While DMADV stands for Design-Measure-Analyze-Design-Verify.

3. **Fact and Data-Based Approach:**

   The statistical and methodical method shows the scientific basis of the technique.

Quality Standard Models: Six-Sigma

- **Characteristics of Six Sigma:** The Characteristics of Six Sigma are as follows:

4. **Project and Objective-Based Focus:**

   The Six Sigma process is implemented to focus on the requirements and conditions.

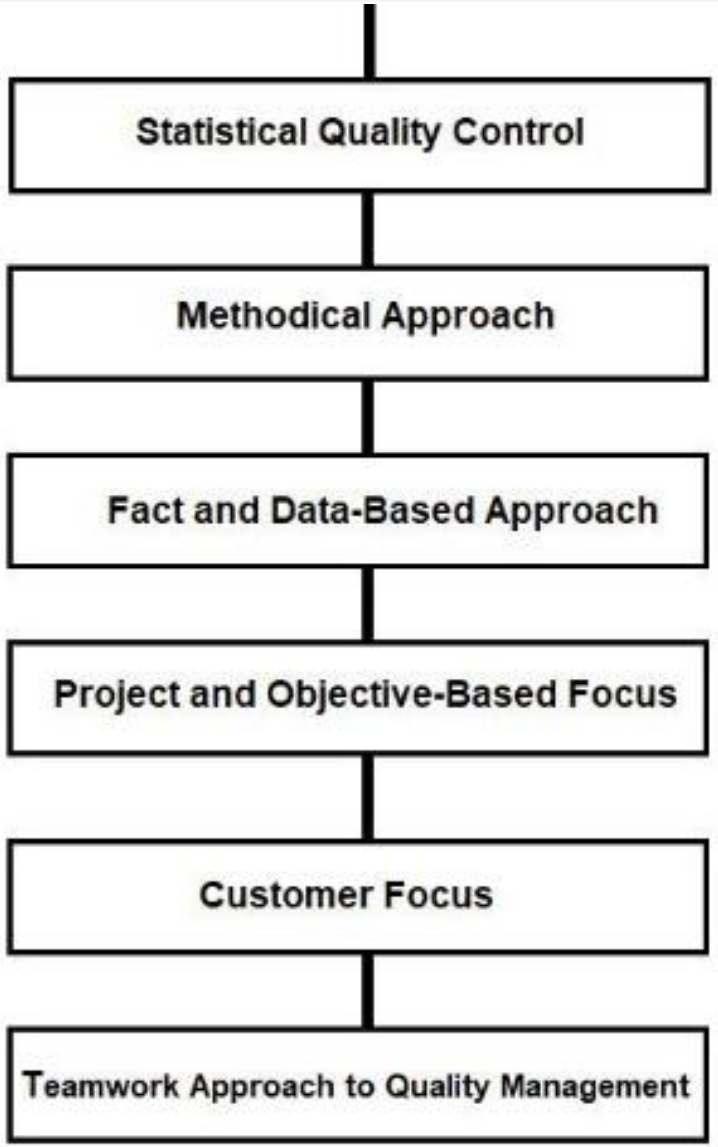
5. **Customer Focus:**

   The customer focus is fundamental to the Six Sigma approach. The quality improvement and control standards are based on specific customer requirements.

6. **Teamwork Approach to Quality Management:**

   The Six Sigma process requires organizations to get organized for improving quality.

Quality Standard Models: Six-Sigma



Quality Standard Models: Six-Sigma

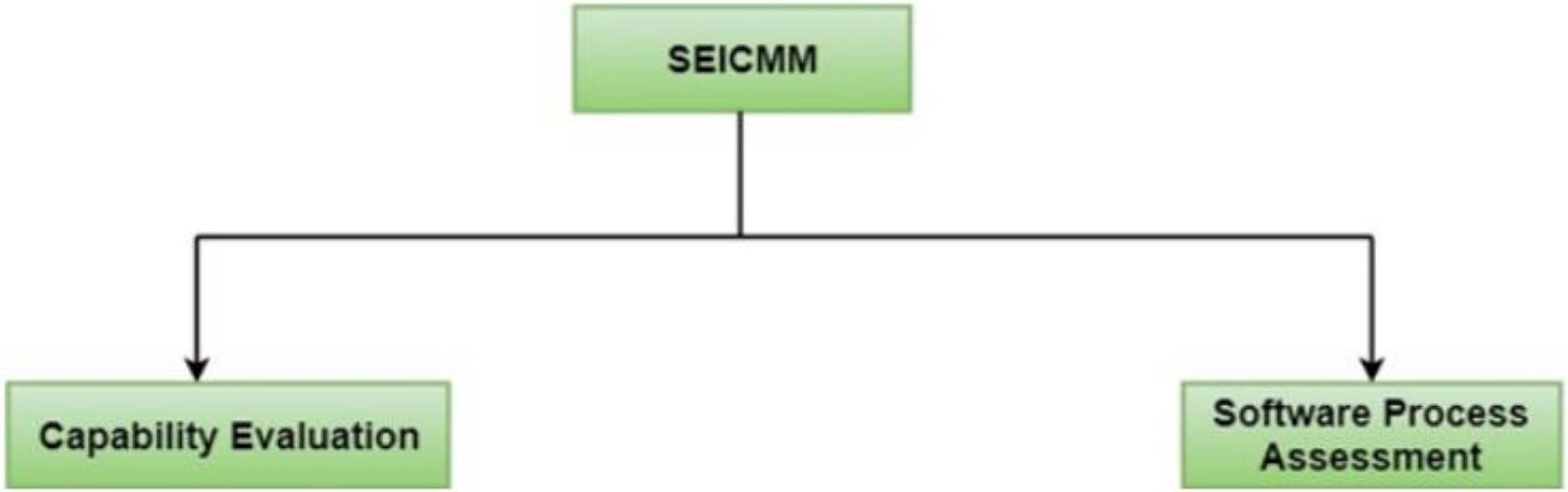
- **Six Sigma Methodologies:** Two methodologies used in the Six Sigma projects are DMAIC and DMADV.
- **DMAIC** is used to enhance an existing business process. The DMAIC project methodology has five phases:
  1. Define
  2. Measure
  3. Analyze
  4. Improve
  5. Control
- **DMADV** is used to create new product designs or process designs. The DMADV project methodology also has five phases:
  1. Define
  2. Measure
  3. Analyze
  4. Design
  5. Verify

Process Improvement Model CMM

- The Capability Maturity Model (CMM) is a procedure used to develop and refine an organization's software development process.
- The model defines a five-level evolutionary stage of increasingly organized and consistently more mature processes.
- CMM was developed and is promoted by the Software Engineering Institute (SEI), a research and development center promote by the U.S. Department of Defense (DOD).
- Capability Maturity Model is used as a benchmark to measure the maturity of an organization's software process.

Process Improvement Model CMM

- There are two methods of SEICMM:



Process Improvement Model CMM

❑Capability Evaluation:

❑Capability evaluation provides a way to assess the software process capability of an organization.

❑The results of capability evaluation indicate the likely contractor performance if the contractor is awarded a work.

❑Therefore, the results of the software process capability assessment can be used to select a contractor.

Process Improvement Model CMM

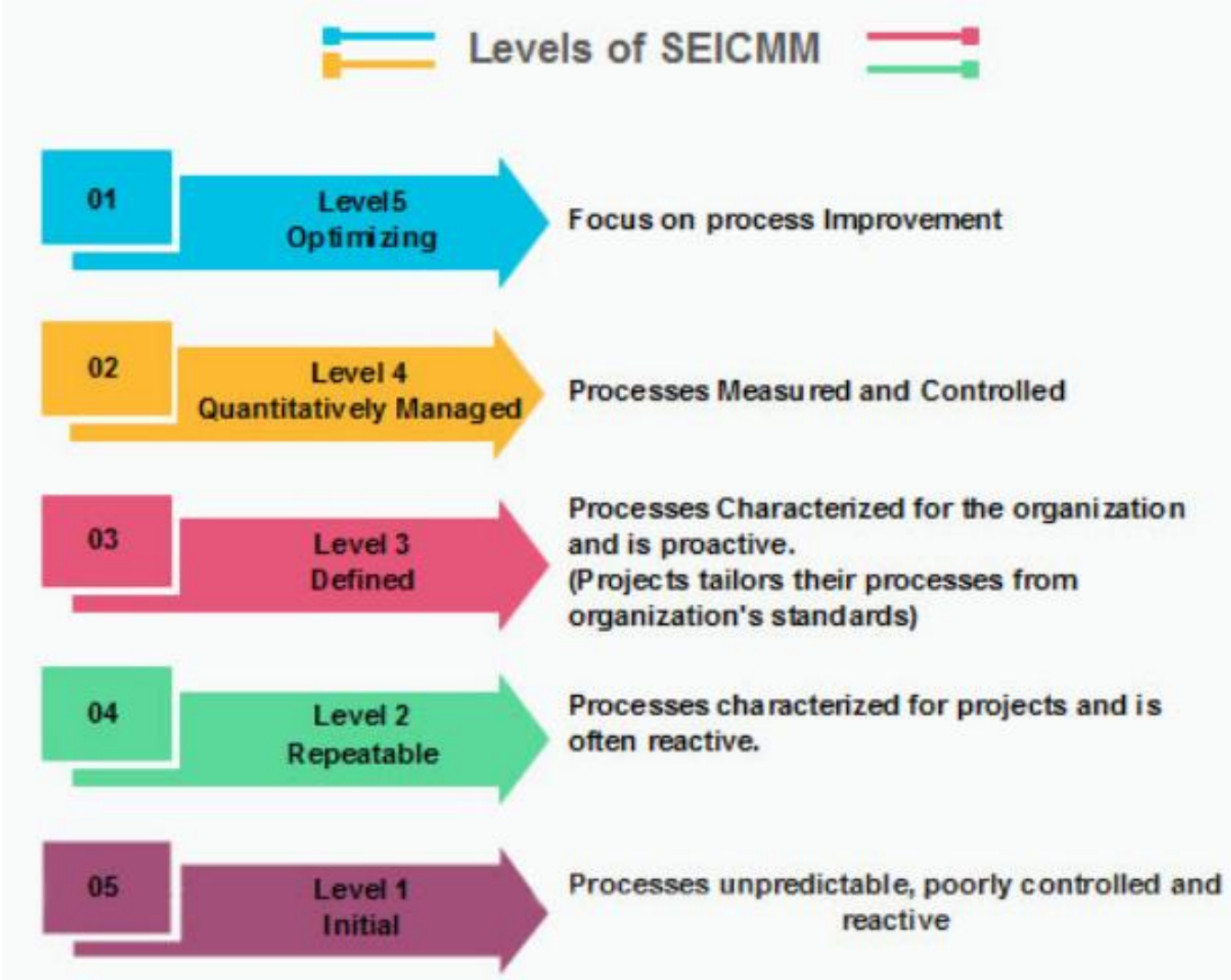
❑Software Process Assessment:

❑Software process assessment is used by an organization to improve its process capability.

❑Thus, this type of evaluation is for purely internal use.

❑SEI CMM categorized software development industries into the following five maturity levels.

❑The various levels of SEI CMM have been designed so that it is easy for an organization to build its quality system starting from scratch slowly.



Process Improvement Model CMM

❑**Software Process Assessment:**

❑**Level 1: Initial:** Ad hoc activities characterize a software development organization at this level. Very few or no processes are described and followed. Since software production processes are not limited, different engineers follow their process and as a result, development efforts become chaotic. Therefore, it is also called a chaotic level.

❑**Level 2: Repeatable:** At this level, the fundamental project management practices like tracking cost and schedule are established. Size and cost estimation methods, like function point analysis, COCOMO, etc. are used.

❑**Level 3: Defined:** At this level, the methods for both management and development activities are defined and documented. There is a common organization-wide understanding of operations, roles, and responsibilities. The ways through defined, the process and product qualities are not measured. ISO 9000 goals at achieving this level.

Process Improvement Model CMM

❑**Software Process Assessment:**

❑**Level 4: Managed:** At this level, the focus is on software metrics. Two kinds of metrics are composed.

Product metrics measure the features of the product being developed, such as its size, reliability, time complexity, understandability, etc.

Process metrics follow the effectiveness of the process being used, such as average defect correction time, productivity, the average number of defects found per hour inspection, the average number of failures detected during testing per LOC, etc.

❑**Level 5: Optimizing:** At this phase, process and product metrics are collected. Process and product measurement data are evaluated for continuous process improvement.

Process Improvement Model CMM

- Key Process Areas (KPA) of a software organization:
- Except for SEI CMM level 1, each maturity level is featured by several Key Process Areas (KPAs) that contains the areas an organization should focus on improving its software process to the next level.
- The focus of each level and the corresponding key process areas are shown

CMM Level	Focus	Key Process Areas
1. Initial	Competent People	NO KPA'S
2. Repeatable	Project Management	Software Project Planning software Configuration Management
3. Defined	Definition of Processes	Process definition Training Program Peer reviews
4. Managed	Product and Process quality	Quantitiative Process Metrics Software Quality Management
5. Optimizing	Continuous Process improvement	Defect Prevention Process change management Technology change management

Process Improvement Model CMMI

- Capability Maturity Model Integration (CMMI) is a successor of CMM and is a more evolved model that incorporates best components of individual disciplines of CMM like Software CMM, Systems Engineering CMM, People CMM, etc.
- Since CMM is a reference model of matured practices in a specific discipline, so it becomes difficult to integrate these disciplines as per the requirements.
- This is why CMMI is used as it allows the integration of multiple disciplines as and when needed.

Process Improvement Model CMMI

- **Objectives of CMMI :**
- Fulfilling customer needs and expectations.
- Value creation for investors/stockholders.
- Market growth is increased.
- Improved quality of products and services.
- Enhanced reputation in Industry.

Process Improvement Model CMMI

❑Staged Representation :

- uses a pre-defined set of process areas to define improvement path.
- provides a sequence of improvements, where each part in the sequence serves as a foundation for the next.
- an improved path is defined by maturity level.
- maturity level describes the maturity of processes in organization.
- Staged CMMI representation allows comparison between different organizations for multiple maturity levels.

❑Continuous Representation :

- allows selection of specific process areas.
- uses capability levels that measures improvement of an individual process area.
- Continuous CMMI representation allows comparison between different organizations on a process-area-by-process-area basis.
- allows organizations to select processes which require more improvement.
- In this representation, order of improvement of various processes can be selected which allows the organizations to meet their objectives and eliminate risks.

Process Improvement Model CMMI

❑**CMMI Model – Maturity Levels :** In CMMI with staged representation, there are five maturity levels described as follows :

❑**Maturity level 1 : Initial**

- processes are poorly managed or controlled.
- unpredictable outcomes of processes involved.
- ad hoc and chaotic approach used.
- No KPAs (Key Process Areas) defined.
- Lowest quality and highest risk.

❑**Maturity level 2 : Managed**

- requirements are managed.
- processes are planned and controlled.
- projects are managed and implemented according to their documented plans.
- This risk involved is lower than Initial level, but still exists.
- Quality is better than Initial level.

Process Improvement Model CMMI

❑**CMMI Model – Maturity Levels :** In CMMI with staged representation, there are five maturity levels described as follows :

❑**Maturity level 3 : Defined**

- processes are well characterized and described using standards, proper procedures, and methods, tools, etc.
- Medium quality and medium risk involved.
- Focus is process standardization.

❑**Maturity level 4 : Quantitatively managed**

- quantitative objectives for process performance and quality are set.
- quantitative objectives are based on customer requirements, organization needs, etc.
- process performance measures are analyzed quantitatively.
- higher quality of processes is achieved.
- lower risk

Process Improvement Model CMMI

❑**CMMI Model – Maturity Levels :** In CMMI with staged representation, there are five maturity levels described as follows :

❑**Maturity level 5 : Optimizing**

- continuous improvement in processes and their performance.
- improvement has to be both incremental and innovative.
- highest quality of processes.
- lowest risk in processes and their performance.

Process Improvement Model CMMI

❑**CMMI Model – Maturity Levels :** In CMMI with staged representation, there are five maturity levels described as follows :

❑**Maturity level 5 : Optimizing**

- continuous improvement in processes and their performance.
- improvement has to be both incremental and innovative.
- highest quality of processes.
- lowest risk in processes and their performance.

- Experiential Learning (Quality Management)
- Complete the topic on or before 30 April 2024.
- If you have any doubt then ask me in the class of 30 April 2024.

Quality Control

- Quality control encompasses a set of software engineering actions that help to ensure that each work product meets its quality goals.
- Models are reviewed to ensure that they are complete and consistent.
- Code may be inspected in order to uncover and correct errors before testing commences.
- A series of testing steps is applied to uncover errors in processing logic, data manipulation, and interface communication.
- 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.

Quality Assurance

- Quality assurance establishes the infrastructure that supports solid software engineering methods, rational project management, and quality control actions—all pivotal if you intend to build high-quality software.
- In addition, quality assurance consists of a set of auditing and reporting functions that assess the effectiveness and completeness of quality control actions.
- The goal of quality assurance is to provide management and technical staff with the data necessary to be informed about product quality, thereby gaining insight and confidence that actions to achieve product quality are working.
- Of course, if the data provided through quality assurance identifies problems, it is management's responsibility to address the problems and apply the necessary resources to resolve quality issues.

Quality Factors

- Even the most jaded software developers will agree that high-quality software is an important goal.
- But how do we define software quality?
- In the most general sense, software quality can be defined as: An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it. There is little question that the preceding definition could be modified or extended and debated endlessly.
- **Three important points for quality factors:**

1. An effective software process establishes the infrastructure that supports any effort at building a high-quality software product.
2. A useful product delivers the content, functions, and features that the end user desires, but as important, it delivers these assets in a reliable, error-free way.
3. By adding value for both the producer and user of a software product, highquality software provides benefits for the software organization and the enduser community.

Quality Factors

❑Garvin's Quality Dimensions:

1. **Performance quality:** Does the software deliver all content, functions, and features that are specified as part of the requirements model in a way that provides value to the end user?
2. **Feature quality:** Does the software provide features that surprise and delight first-time end users?
3. Reliability: Does the software deliver all features and capability without failure? Is it available when it is needed? Does it deliver functionality that is error-free?
4. **Conformance:** Does the software conform to local and external software standards that are relevant to the application? Does it conform to de facto design and coding conventions?
5. **Durability:** Can the software be maintained (changed) or corrected (debugged) without the inadvertent generation of unintended side effects?
6. **Serviceability:** Can the software be maintained (changed) or corrected (debugged) in an acceptably short time period?
7. **Aesthetics:** There's no question that each of us has a different and very subjective vision of what is aesthetic. Perception: In some situations, you have a set of prejudices that will influence your perception of quality.

Quality Factors

❑McCall's Quality Factors

1. **Correctness:** The extent to which a program satisfies its specification and fulfills the customer's mission objectives.
2. **Reliability:** The extent to which a program can be expected to perform its intended function with required precision.
3. **Efficiency:** The amount of computing resources and code required by a program to perform its function.
4. **Integrity:** Extent to which access to software or data by unauthorized persons can be controlled.
5. **Usability:** Effort required to learn, operate, prepare input for, and interpret output of a program.
6. **Maintainability:** Effort required to locate and fix an error in a program.
7. **Flexibility:** Effort required to modify an operational program.
8. **Testability:** Effort required to test 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—related to the packaging and scope of the functions that the program performs.
11. **Interoperability:** Effort required to couple one system to another.

Quality Factors

❑ISO 9126 Quality Factors

1. **Functionality:** The degree to which the software satisfies stated needs as indicated by the following subattributes: suitability, accuracy, interoperability, compliance, and security.
2. **Reliability:** The amount of time that the software is available for use as indicated by the following subattributes: maturity, fault tolerance, recoverability.
3. **Usability:** The degree to which the software is easy to use as indicated by the following subattributes: understandability, learnability, operability.
4. **Efficiency:** The degree to which the software makes optimal use of system resources as indicated by the following subattributes: time behavior, resource behavior.
5. **Maintainability:** The ease with which repair may be made to the software as indicated by the following subattributes: analyzability, changeability, stability, testability.
6. **Portability:** The ease with which the software can be transposed from one environment to another as indicated by the following subattributes: adaptability, installability, conformance, replaceability.

Methods of Quality Management

❑Software quality doesn't just appear. It is the result of good project management and solid software engineering practice.

❑Management and practice are applied within the context of four broad activities that help a software team achieve high software quality:

1. **Software engineering methods**
2. **Project management techniques**
3. **Quality control actions**
4. **Software quality assurance.**

Methods of Quality Management

❑**Software Engineering Methods:** If you expect to build high-quality software, you must understand the problem to be solved. You must also be capable of creating a design that conforms to the problem while at the same time exhibiting characteristics that lead to software that exhibits the quality dimensions and factors.

❑**Project Management Techniques:** The impact of poor management decisions on software quality given the following implications : if (1) a project manager uses estimation to verify that delivery dates are achievable, (2) schedule dependencies are understood and the team resists the temptation to use short cuts, (3) risk planning is conducted so problems do not breed chaos, software quality will be affected in a positive way.

Methods of Quality Management

❑**Quality Control:** encompasses a set of software engineering actions that help to ensure that each work product meets its quality goals. Models are reviewed to ensure that they are complete and consistent. Code may be inspected in order to uncover and correct errors before testing commences. A series of testing steps is applied to uncover errors in processing logic, data manipulation, and interface communication. 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.

❑**Quality Assurance:** establishes the infrastructure that supports solid software engineering methods, rational project management, and quality control actions—all pivotal if you intend to build high-quality software. In addition, quality assurance consists of a set of auditing and reporting functions that assess the effectiveness and completeness of quality control actions. The goal of quality assurance is to provide management and technical staff with the data necessary to be informed about product quality, thereby gaining insight and confidence that actions to achieve product quality are working.

Note for Students

- This power point presentation is for lecture, therefore it is suggested that also utilize the text books and lecture notes.