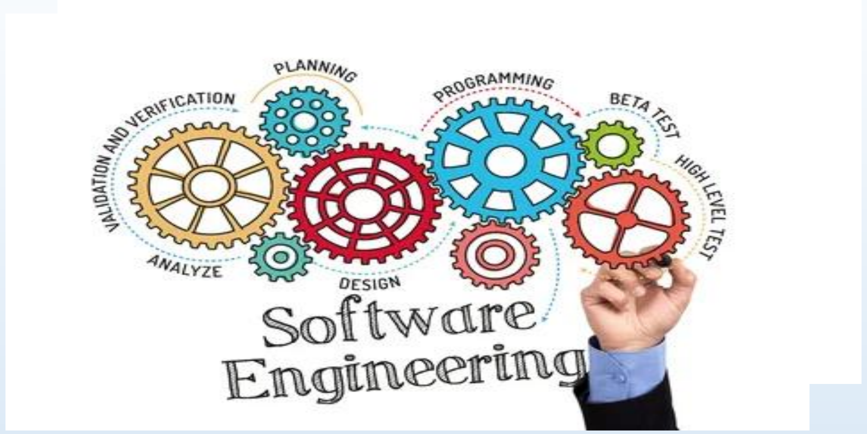
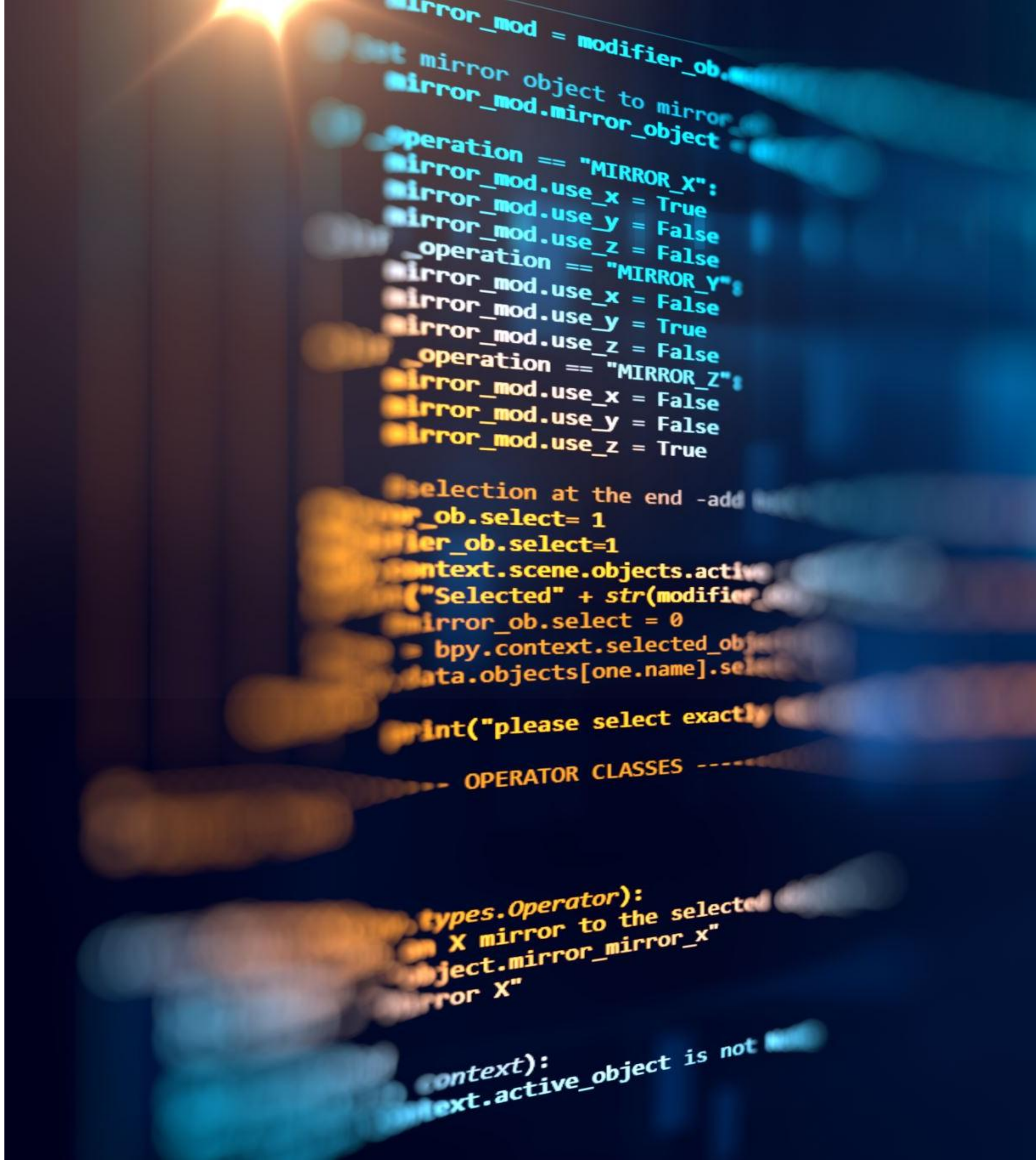
INFORMATION SYSTEMS 3A

Software Quality Management



Contents

- Software Quality
- Principles concerns of software quality
- Quality Plans
- Non-functional requirements to assess the quality of software
- Factors Affecting Software Quality
- Software Quality Metrics
- Software Quality Management
- How to Achieve Software Quality?
- Quality Assurance Vs Quality Control



```
irror_mod = modifier_ob.
 mirror object to mirror
irror_mod.mirror_object
peration == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
_operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
_operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True

selection at the end -add
_ob.select= 1
ier_ob.select=1
ntext.scene.objects.acti
("Selected" + str(modifie
irror_ob.select = 0
 bpy.context.selected_ob
ata.objects[one.name].se

int("please select exactl

- OPERATOR CLASSES ----

types.Operator):
 X mirror to the selecte
ject.mirror_mirror_x"
rror X"

ontext):
ext.active_object is not
```

Software Quality

• The quality of software can be defined as the ability of the software to function as per user requirements.

• When it comes to software products, they must satisfy all the functionalities written down in the SRS document.

Software Quality includes:

-**Good Design:** Good visualization design to attract users.

-**Durability:** The software works without any issue for a long period of time.

-**Consistency:** Software performs consistently on different platforms and other devices.

-**Maintainability:** Capture and fix bugs quickly. New features are added easily.

-**Value for money:** Customers & companies who make this app should feel that the money spent on this app was not wasted.

Three principles concerns of software quality

- **At the organizational level**, quality management is concerned with establishing a framework of organizational processes and standards that will lead to high-quality software.
- **At the project level,** quality management involves the application of specific quality processes and checking that these planned processes have been followed.
- **At the project level,** quality management is also concerned with establishing a quality plan for a project. The quality plan should set out the quality goals for the project and define what processes and standards are to be used.

Quality plans

- A **quality plan** is a document that outlines the strategy, scope, processes, resources, and goals for ensuring that software products meet the required quality standards. It acts as a roadmap for the entire software development and testing process to ensure that the software delivered to customers meets both functional and non-functional requirements, is reliable and secure, and performs as expected.
- Quality plan structure
  - Product introduction
  - Product plans
  - Process descriptions
  - Quality goals
  - Risks and risk management
- Quality plans should be short, succinct(Compact and Precise) documents
  - If they are too long, no one will read them.

Product introduction

- This section provides an overview of the product or service being developed or delivered.
- It typically includes the product’s purpose, key features, specifications, and target market.
- The introduction sets the context for the quality management process, emphasizing how quality is essential to meeting the product's objectives and customer expectations.

Product plans

- Product plans outline the roadmap for the development, testing, and delivery of the product.
- It includes detailed timelines, milestones, and the resources required for successful product realization.
- This section focuses on defining the stages of product development, such as design, prototype, production, and launch, while ensuring quality is integrated at every stage

Process descriptions

- This section provides detailed descriptions of the processes involved in product development and production.
- It typically includes process flows, standard operating procedures (SOPs), and instructions for how tasks should be performed.
- The goal is to ensure that the processes are clearly defined, repeatable, and efficient, with a focus on consistent quality control throughout.

Quality goals

- Quality goals are the measurable objectives that the organization aims to achieve during the product's lifecycle.
- These goals could be related to product performance, customer satisfaction, defect rates, compliance, or other relevant metrics. The quality goals should be aligned with both the organization’s strategic objectives and customer requirements.
- They act as benchmarks for measuring and driving quality improvements.

Risks and risk management

- This section identifies potential risks that could impact product quality, timelines, or customer satisfaction.
- Risks can be related to technical challenges, resource limitations, market fluctuations, or regulatory compliance.
- The risk management plan outlines how each risk will be assessed, prioritized, and mitigated.
- This includes specifying risk owners, defining mitigation strategies, and monitoring risks throughout the product development lifecycle to ensure that quality standards are maintained.

Non-functional requirements to assess quality of software

- **Maintainability:** The ease with which software can be modified (adding features, enhancing features, fixing bugs, etc.)
- **Portability:** The ability of software to be transferred easily from one location to another.
- **Functionality:** The ability of software to carry out the functions as specified or desired.
- **Performance:** The speed at which software performs under a particular load.
- **Compatibility:** The suitability of software for use in different environments like different devices, operating systems, and browsers.
- **Usability:** The degree of software's ease of use.
- **Reliability:** The ability of software to perform a required function under stated conditions without any errors.
- **Security:** The extent of protection of software against unauthorized access, invasion of privacy, theft. loss of data etc. Ex. OTP

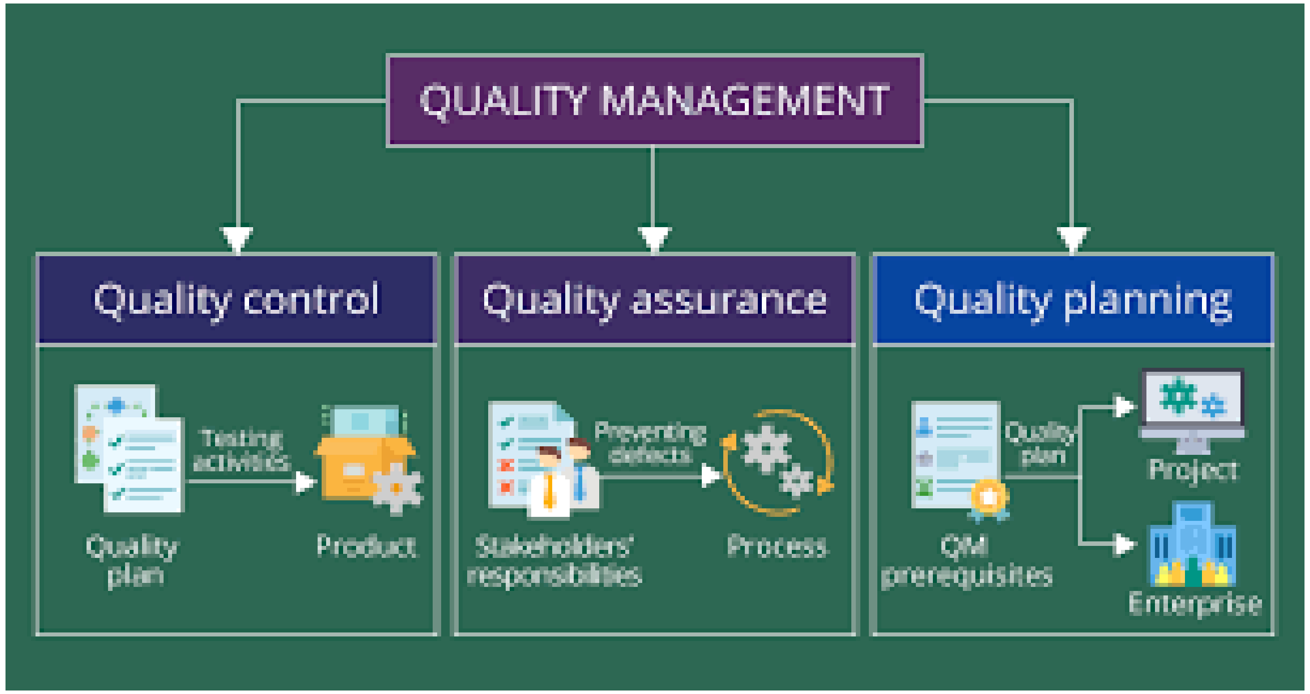
Software Quality Management

- Software Quality Management (SQM) refers to the complete process that ensures a software product is developed as per national and international standards like ANSI, IEEE, and ISO.

  Need for Software Quality Management:

  1. Deliver high-quality products on time.
  2. Increases stakeholder faith in product & company.
  3. High-quality products always ensure customer satisfaction.

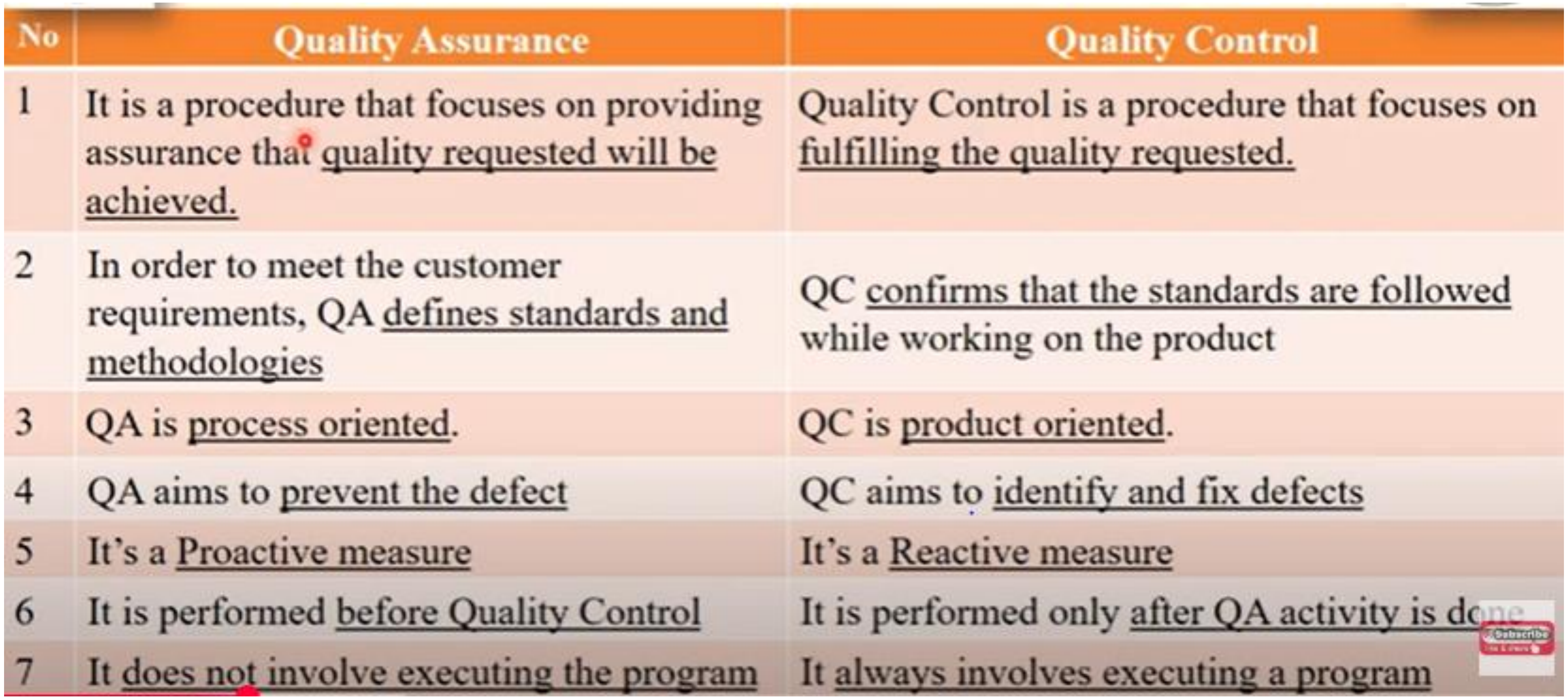
Activities of Software Quality Management



How to Achieve Software Quality?

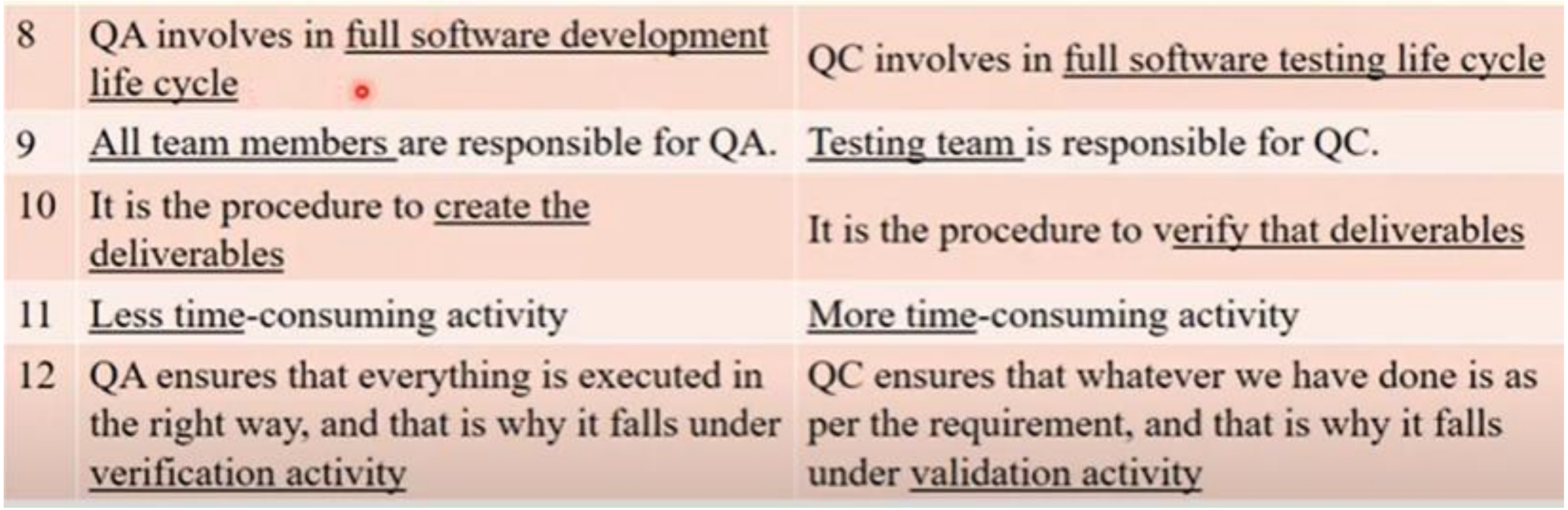
- **Quality Planning:**
  - Select applicable procedures and standards for a particular project and modify them as required to develop a quality plan.
- **Quality Assurance:**
  - It assures that the system meets specified requirements and customer expectations.
  - It defines standards and methodologies for a successful development process.
  - It assures Correctness, Efficiency, Flexibility, Maintainability, Portability, Usability, etc.
- **Quality Control:**
  - It focuses on achieving & fulfilling quality parameters or quality goals as per customer requirements.
  - It focuses on delivering products on time with accurate cost.

Quality Assurance Vs Quality Control



No	Quality Assurance	Quality Control
1	It is a procedure that focuses on providing assurance that quality requested will be achieved.	Quality Control is a procedure that focuses on fulfilling the quality requested.
2	In order to meet the customer requirements, QA defines standards and methodologies	QC confirms that the standards are followed while working on the product
3	QA is process oriented.	QC is product oriented.
4	QA aims to prevent the defect	QC aims to identify and fix defects
5	It's a Proactive measure	It's a Reactive measure
6	It is performed before Quality Control	It is performed only after QA activity is done
7	It does not involve executing the program	It always involves executing a program

Quality Assurance Vs Quality Control



8	QA involves in full software development life cycle	QC involves in full software testing life cycle
9	All team members are responsible for QA.	Testing team is responsible for QC.
10	It is the procedure to create the deliverables	It is the procedure to verify that deliverables
11	Less time-consuming activity	More time-consuming activity
12	QA ensures that everything is executed in the right way, and that is why it falls under verification activity	QC ensures that whatever we have done is as per the requirement, and that is why it falls under validation activity

Software Quality Metrics

Software Quality Management makes sure that the software is high-quality and follows proper rules and standards. To check how good the software is, we use different types of metrics (measurements):

Customer Problem Metrics:

- Measure the problems encountered by the customers while using the product.
- Problems per User Month = (Total Number of Problem Reports) / (Total Number of Users * Number of Months)

- For example, if you have 10 problem reports, 100 users, and you're tracking data for 1 month, the problems per user month would be 10 / (100 * 1) = 0.1.

Customer Satisfaction Metrics:

- It deals with the overall quality of the product & how much a customer is satisfied with that product.
- It is measured by Very Satisfied, Satisfied, Neutral, Dissatisfied, and Very Dissatisfied.

Software Maintenance Metrics:

- After completion of Development & Testing product release in the market.
- During this interval. How many defect arrived at customer environment?

Reviews and inspections

- A group examines part or all of a process or system and its documentation to find potential problems.
- Software or documents may be 'signed off' at a review, which signifies that progress to the next development stage has been approved by management.
- There are different types of reviews with different objectives
  - Quality reviews (product and standards)
  - Inspections for defect removal (product)
  - Reviews for progress assessment (product and process)

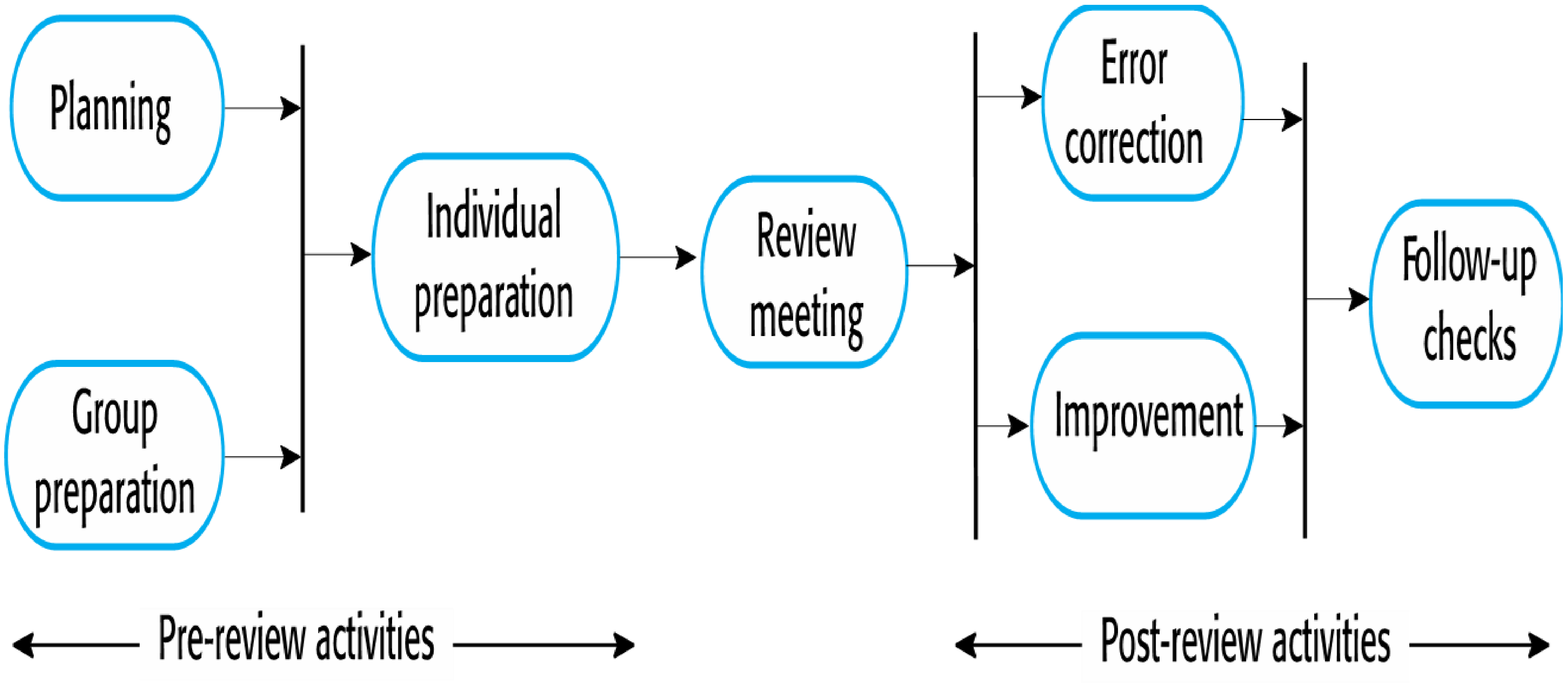
Quality reviews

- A group of people carefully examines part or all of a software system and its associated documentation.
- Code, designs, specifications, test plans, standards, etc., can all be reviewed.
- Software or documents may be 'signed off' at a review, which signifies that progress to the next development stage has been approved by management.

Phases in the review process

- Pre-review activities
  - Pre-review activities are concerned with review planning and review preparation
- The review meeting
  - During the review meeting, an author of the document or program being reviewed should ‘walk through’ the document with the review team.
- Post-review activities
  - These address the problems and issues that have been raised during the review meeting.

The software review process



Program inspections

- These are peer reviews where engineers examine the source of a system with the aim of discovering anomalies and defects.
- Inspections do not require execution of a system so may be used before implementation.
- They may be applied to any representation of the system (requirements, design, configuration data, test data, etc.).
- They have been shown to be an effective technique for discovering program errors.

Inspection checklists

- Checklist of common errors should be used to drive the inspection.
- Error checklists are programming language dependent and reflect the characteristic errors that are likely to arise in the language.
- In general, the 'weaker' the type checking, the larger the checklist.
- Examples: Initialisation, Constant naming, loop termination, array bounds, etc.

An inspection checklist (a)

Fault class	Inspection check
Data faults	• Are all program variables initialized before their values are used? • Have all constants been named? • Should the upper bound of arrays be equal to the size of the array or Size -1? • If character strings are used, is a delimiter explicitly assigned? • Is there any possibility of buffer overflow?
Control faults	• For each conditional statement, is the condition correct? • Is each loop certain to terminate? • Are compound statements correctly bracketed? • In case statements, are all possible cases accounted for? • If a break is required after each case in case statements, has it been included?
Input/output faults	• Are all input variables used? • Are all output variables assigned a value before they are output? • Can unexpected inputs cause corruption?

An inspection checklist (b)

Fault class	Inspection check
Interface faults	• Do all function and method calls have the correct number of parameters? • Do formal and actual parameter types match? • Are the parameters in the right order? • If components access shared memory, do they have the same model of the shared memory structure?
Storage management faults	• If a linked structure is modified, have all links been correctly reassigned? • If dynamic storage is used, has space been allocated correctly? • Is space explicitly deallocated after it is no longer required?
Exception management faults	• Have all possible error conditions been taken into account?