**Quiz -1**

**Chapter-1**

### ****1. What is Software?****

**Software** is more than just a program. It includes:

* Programs (what the computer runs)
* Procedures (how to run them)
* Documentation and data

#### **TYPES:**

* **Generic (Buy):** Ready-made. Same for everyone.  
  ✅ **Example:** MS Word, Google Chrome
* **Customized (Build):** Made for a specific user or business.  
  ✅ **Example:** A hospital management system made only for a specific hospital

### ****2. What is Software Quality?****

* **ISO/IEC 9126:** Software quality is how well a software meets users’ needs.
* **IEEE Std 610:** It's how well software meets **requirements and user expectations.**

✅ **Example:** A banking app that works fast, safely, and without errors = high quality.

### ****3. Challenges in Software Projects****

3 main challenges:

* **Time:** Late delivery
* **Scope:** Low quality, missing features
* **Cost:** Too expensive

✅ **Example:** A mobile app that is released 3 months late and still crashes often has these issues.

### ****4. Examples of Software Defects****

* **Therac-25 (1986):** Radiation machine killed people due to software bug.
* **Ariane 5 Rocket (1996):** Exploded 37 seconds after launch because of software error.

✅ **Lesson:** Software bugs can be dangerous and costly.

### ****5. What is Software Testing?****

Software Testing = Running the software to **find bugs** and **check requirements**.

**Goals of Testing:**

* Find mistakes
* Check performance
* Make sure system works correctly

✅ **Example:** Testing an autopilot system in an airplane to make sure it follows flight rules.

### ****6. Levels of Testing****

* **Unit Testing:** Test small parts/modules  
  ✅ Example: Testing a login form separately
* **Integration Testing:** Combine modules and test together  
  ✅ Example: Login + dashboard
* **System Testing:** Test full system  
  ✅ Example: Full app testing on phone with real data

### ****7. Role of Testing****

* **Static Analysis (manual):** Check code without running it  
  ✅ Example: Code review
* **Dynamic Analysis (automated):** Run the software and test  
  ✅ Example: Using Selenium to test website clicks

Both methods are useful together.

### ****8. What is Software Quality Assurance (SQA)?****

SQA = A planned way to **ensure software is good quality** during the whole process.

✅ **Example:** Making a checklist for developers to follow good coding standards.

### ****9. What is Software Quality Control (SQC)?****

SQC = Actually **checking** the final product to make sure it meets the plan.

✅ **Example:** Reviewing design, code, and test plan before delivering to the client.

### ****10. Difference Between SQA and SQC****

| **SQA** | **SQC** |
| --- | --- |
| Focus on **process** | Focus on **product** |
| Happens **throughout development** | Happens **after development** |
| **Prevention** | **Detection** |

✅ Example:  
SQA = Use coding rules from the start  
SQC = Test the app before release

### ****11. Validation and Verification****

* **Verification:** “Did we build the software **right**?” (Check against requirements)
* **Validation:** “Did we build the **right** software?” (Check if user is happy)

✅ Example:  
Verification = Review login logic in code  
Validation = Ask user if login feature works as expected

### ****12. Difficulties in Quality Expectations****

* **Size:** Too much code = hard to test
* **Complexity:** Logical bugs are hard to find
* **Environment:** Software might break under stress
* **Flexibility:** Users want constant updates
* **Cost:** Good testing tools are expensive

✅ Example: Testing a result website on result day when millions try to access it

### ****13. Software Quality Engineering (SqE)****

Ensures quality using:

* **Planning** QA activities
* **Execution** of tests
* **Decision** making on pass/fail
* **Measurement** of quality results

✅ **Example:** A software company tests all features, logs bugs, and checks if app meets quality targets.

### ****14. SqE Activities****

* **Testing**: Find and fix bugs
* **Defect Prevention**: Plan to avoid bugs
* **Formal Verification**: Reviews, walkthroughs
* **Fault Tolerance**: Prepare for failure (like auto-save)

✅ Example: Word auto-saves your work in case of a crash

### ****15. Error, Fault, Failure, and Defect****

* **Error:** Human mistake  
  ✅ Example: Typing wrong formula in code
* **Fault:** Bug in the code  
  ✅ Example: Incorrect login logic
* **Failure:** Software doesn't work  
  ✅ Example: App crashes during login
* **Defect:** General name for all above  
  ✅ Example: Any unexpected behavior

### ****16. Complete Testing****

**Complete Testing = Test everything.** But it's **impossible** because:

* Too many possible inputs
* Too complex designs
* Too many environments

✅ Example: Testing every possible GPS path in a self-driving car is not possible

### ****17. Testing Activities****

Steps:

1. Choose what to test
2. Pick inputs
3. Predict output
4. Set up system
5. Run test
6. Compare result with expected

✅ Example:  
Input: 2 + 2  
Expected: 4  
System says 5 → Bug found

* **Co-existence**: Runs well with other apps.  
  *Example: Antivirus doesn’t slow down other programs.*
* **Replaceability**: Can replace another software.  
  *Example: Google Sheets can replace Excel.*
* **Compliance**: Meets portability standards.  
  *Example: Software that runs on both Windows and Linux.*

**Chapter-2**

**1. Quality Perspectives**

Different people see software quality in different ways:

* **External (Customer/User):** They care about the behavior of the software — if it works well or not.  
  ✅ *Example:* A student using a university result app — they care if it shows results correctly.
* **Internal (Developer/Testers):** They look inside the software — how it's built and works.  
  ✅ *Example:* A developer checks if the code is clean and follows rules.
* **Others (3rd party):** Like systems sending email alerts or external services.  
  ✅ *Example:* A notification system that emails students after result publication.

**2. External vs Internal Views**

* **External View = Black Box:** You only see what the software does (not how it works).  
  ✅ *Example:* Clicking a button in an app and seeing the result.
* **Internal View = White/Clear Box:** You can see how everything works inside.  
  ✅ *Example:* A developer checking how clicking that button runs the code.

**3. Views in Software Quality (Kitchenham & Pfleeger)**

1. **Mystical View** – “I know it’s good when I see it.” Not measurable.  
   ✅ *Example:* Saying “this app just feels right” without clear reason.
2. **User View** – Quality means it meets user needs.  
   ✅ *Example:* A website is good if students can easily apply to universities.
3. **Manufacturing View** – Software must follow rules and be done right the first time.  
   ✅ *Example:* A team follows ISO standards to write bug-free code.
4. **Product View** – If inside is built well, outside will work well.  
   ✅ *Example:* Using good coding practices leads to a better app.
5. **Value-Based View** – Quality = Excellence + Worth.  
   ✅ *Example:* A company pays more for software that is fast and secure.

**4. Why Measure Quality?**

To get numbers and facts, not just opinions.

* **Baseline:** Set a target  
  ✅ *Example:* Users should find info on a website in under 20 minutes
* **Improvement:** See what to fix  
  ✅ *Example:* Reduce app crashes after measuring user complaints
* **Planning:** Know current level to plan better  
  ✅ *Example:* Know how fast your website loads before trying to improve it.

**5. Software Quality Factors**

A quality factor represents the behavioral characteristic of a system. These are the main qualities of good software:

**🔒 Integrity/Security**

Control access to data  
✅ *Example:* Only auditors can see user bank history.

**⚡ Efficiency**

How much system resource is used  
✅ *Example:* App should use only 75% of CPU during heavy use.

**👩‍💻 Usability**

How easy it is to use  
✅ *Example:* Student should complete university form in 4–6 minutes.

**✔ Reliability & Correctness**

Software works as expected without failure  
✅ *Example:* Only 5 out of 1000 runs can fail due to software issues.

**6. Software Quality Criteria**

These are technical things to help reach the quality factors:

**🧱 Modularity**

Software is made of small parts/modules  
✅ *Example:* If a module crashes, others can still work.

**🔧 Maintainability**

How easy it is to fix issues  
✅ *Example:* Fixing a report format should take less than 20 hours.

**✅ Testability**

How easy it is to test  
✅ *Example:* Code should not be too complex (cyclomatic complexity ≤ 20).

**🔄 Flexibility**

How easy it is to make changes  
✅ *Example:* An experienced programmer can change output settings in 1 hour.

**🛫 Portability**

Can move software to other systems  
✅ *Example:* Run the app on both Windows and Mac.

**♻ Reusability**

Parts can be used in other systems  
✅ *Example:* Chemical input form reused in other lab software.

**🔗 Interoperability**

Can work with other systems  
✅ *Example:* App can import files from ChemiDraw.

**7. ISO-9126 Quality Framework**

A well-known standard for software quality. It includes 6 key characteristics:

1. **Functionality** – Does what it’s supposed to do  
   ✅ *Example:* A calculator app gives correct results
2. **Reliability** – Keeps working correctly  
   ✅ *Example:* A banking app never crashes while transferring money
3. **Usability** – Easy to use  
   ✅ *Example:* A food delivery app that’s simple for everyone
4. **Efficiency** – Uses system resources well  
   ✅ *Example:* Doesn’t lag or freeze
5. **Maintainability** – Easy to fix or upgrade  
   ✅ *Example:* A coder can update a feature in less than a day
6. **Portability** – Works on different devices  
   ✅ *Example:* Same app works on both Android and iOS

**8. Quality Expectations**

**External (Consumer) Side**

* **“Good enough” for the price**
* **Fit-for-use** = Does the right thing
* **Conformance** = Done the right way

✅ *Example:* A budget phone app may not be perfect but should still work well

**Internal (Producer) Side**

* **Good enough for the cost**
* Focuses on V&V (Validation & Verification), Maintainability, Interoperability, Modularity

✅ *Example:* Developers build software that's easy to test, update, and connect with others