🌟 QUALITY PERSPECTIVES (What people think about software quality)

🎯 Who is involved in judging software quality? (Subject of SQAT)

1. **External/Consumer** – People who **use** the software (like customers or end-users).
2. **Internal/Producer** – People who **build** the software (like developers, testers, managers).
3. **Others (3rd Party)** – People who **indirectly use** the software (like someone who gets email notifications from a system).

🧱 What is judged? (Objects of SQAT)

- Software **products**, **systems**, and **services**.

👀 Different Views of Software

🖤 External View

- Think of software like a **black box**.
- You can **see what it does**, but you **can't see inside** how it works.
- Example: You use a calculator app and just care about the result.

🤍 Internal View

- Think of software like a **white/clear box**.
- You can **see how it works inside** (code, logic, design).
- Example: A developer looks at the code behind that calculator app.

📚 VIEWS IN SOFTWARE QUALITY (Kitchenham and Pfleeger)

1. ✨ Mystical View (Misconception)

- People believe **"I know good quality when I see it"**, but they **can't explain why**.
- It's like saying "this software just *feels* good".

2. 🤷 User View

- Focuses on the **user's needs and satisfaction**.
- If users are happy and the software works as expected, then it's good quality.
- This includes things like:
    - **Usability** (easy to use)

- **Reliability** (doesn't crash)
- **Efficiency** (fast and smooth)

3. 🏭 Manufacturing View

- Focuses on **following standards and requirements** during development.
- If the software is made exactly as planned, it's considered high quality.
- Example: No bugs, no need to fix things later.
- It's about doing things **right the first time**.
- Models like **CMM** and **ISO 9001** follow this view.

4. 📦 Product View

- If the **inside parts (code, design)** are good, then the **outside performance** will also be good.
- Quality comes from **building it well**.

5. 💰 Value-Based View

- Quality depends on **how much customers are willing to pay** for it.
- Combines:
  - **Excellence** (how good it is)
  - **Worth** (how valuable it is to the user)
- Sometimes a **balance** is needed between **cost** and **quality**.

💉 WHY MEASURE QUALITY? (In simple words)

🧠 Why should we measure software quality using numbers?

Because numbers help us **see clearly** how good the software is. Instead of just saying "it feels good," we can say exactly **how good** it is.

🔑 Main Reasons to Measure Quality

1. 📊 **Baseline (Set a standard)**
   - Measurement helps us decide what level of quality we want.

- Example: "Users should be able to find all needed info from the website in 20 minutes."
- This becomes a **goal** to reach.

2. 💰 **Quality improvement based on cost**
   - Improving quality (like fixing bugs or making the app faster) usually **costs money**.
   - We use measurement to see **what's worth improving** and where to spend money.
   - Helps make **smart decisions**.
3. 🗓️ **Know current status to plan future**
   - We need to **know where we stand now** before planning for the future.
   - Example: If we measure and see that usability is weak, we can plan to improve that part in the next version.

🧩 What is a Software Quality Factor?

A **quality factor** is a special feature or behavior that shows **how good or bad a software is**.
It helps us understand how well the software performs, how easy it is to use, how safe it is, and so on.

👉 You can think of it like **marks** given to a software based on different subjects (like speed, security, correctness, etc.).

📚 Main Software Quality Factors (Explained Simply)

1️⃣ Correctness

✅ What it means:

Correctness means the software does the **right thing** – exactly what it's supposed to do.

💬 In easy words:

If you give the right input, it should give you the correct output every time.

🎯 Example:

If you use a calculator app and do 5 + 7, it must give you 12. If it gives something wrong, then it's not correct.

2️⃣ Reliability

✅ **What it means:**

Reliability means the software works **consistently** and doesn't fail often.

💬 **In easy words:**

You can **trust** the software to work properly for a long time without crashing or giving errors.

🎯 **Example:**

An exam result system should show results properly every time. If it crashes or gives wrong results sometimes, it’s not reliable.

3️⃣ Efficiency

✅ **What it means:**

Efficiency means the software uses **less computer resources** (CPU, memory, etc.) and still works well.

💬 **In easy words:**

It does the job quickly and smoothly without slowing down the device or draining the battery.

🎯 **Example:**

A video editing app that works well even on a low-end laptop and doesn’t use too much RAM is efficient.

4️⃣ Performance

✅ **What it means:**

Performance is about how **fast and responsive** the software is.

💬 **In easy words:**

The software should not be slow – it should open fast and respond quickly to user actions.

🎯 **Example:**

A food delivery app should load restaurants in 2–3 seconds, even when many users are using it at once.

5️⃣ Security / Integrity

✅ **What it means:**

Security means protecting the software and data from **unauthorized people** or hackers.

💬 **In easy words:**

Only the right people can access certain features or data, and others are blocked.

🎯 **Example:**

In a banking app, only the account owner should be able to see their balance. No one else should be able to open that info.

6️⃣ Usability

✅ **What it means:**

Usability means how **easy it is to learn and use** the software.

💬 **In easy words:**

Even if someone is using the software for the first time, they should understand what to do without confusion.

🎯 **Example:**

A website for university applications should be so simple that students can find and submit their forms easily, within 30 minutes.

📄 **Quick Table Recap – Software Quality Factors**

Quality Factor	What it Means	Easy Example
Correctness	Gives correct results	Calculator shows 5 + 7 = 12
Reliability	Works properly without errors	Exam result system doesn't crash
Efficiency	Uses fewer resources	Video editor runs well on weak PC
Performance	Fast and responsive	Food app loads restaurants in 2–3 seconds
Security	Protects from unauthorized access	Only bank user can view their balance
Usability	Easy to use and understand	Students find application info on website easily

📚 What is a Software Quality Criterion?

A **quality criterion** is like a small part or a feature that helps support a bigger **quality factor**.
These are specific features that make the software easier to build, test, maintain, or reuse.

👉 You can think of **quality factors** as the main goals (like **Correctness**, **Usability**)
And **quality criteria** as the tools or ingredients used to reach those goals.

📚 Software Quality Criteria (Explained Simply)

1️⃣ Modularity

✅ **What it means:**

Dividing the software into **smaller, separate parts (modules)** that each do one specific job.

💬 **In easy words:**

If each part of your software is kept in different boxes (modules), it's easier to build, fix, or improve.

🎯 **Example:**

In a shopping app, there can be separate modules for login, product listing, cart, and payment.

2️⃣ Maintainability

✅ **What it means:**

How easy it is to **find and fix problems** in the software after it's released.

💬 **In easy words:**

If a bug appears, how quickly can a developer find it and fix it?

🎯 **Example:**

If government rules change, a developer should be able to update the chemical report feature in **20 hours or less**.

3️⃣ Testability

✅ **What it means:**

How easily we can **test the software** to make sure it works properly.

💬 **In easy words:**

The code should be written in such a way that it's easy to test if everything works correctly.

🎯 **Example:**

If a software module has **low cyclomatic complexity** (less decision-making steps), it will be easier to test.

4️⃣ Flexibility

✅ **What it means:**

How easy it is to **change or upgrade** the software without breaking things.

💬 **In easy words:**

Software should be easy to update or add new features to.

🎯 **Example:**

A developer should be able to add a **new print feature** in just 1 hour of work if the code is flexible.

5️⃣ Portability

✅ **What it means:**

How easy it is to **move the software** to another computer system or platform.

💬 **In easy words:**

The software should run well on **different types of machines or systems** without needing full rewrites.

🎯 **Example:**

If your software works on Windows, it should be easy to make it work on Linux too.

6️⃣ Reusability

✅ **What it means:**

How easily **parts of the software** can be used in **other projects**.

💬 **In easy words:**

Instead of building something from scratch again, we can reuse the already-made part.

🎯 **Example:**

The chemical structure input function should be reusable in other software that also uses standard chemical structures.

7️⃣ Interoperability

✅ **What it means:**

How easily the software can **connect and work with other software** or systems.

💬 **In easy words:**

Different apps or systems should be able to share data and work together.

🎯 **Example:**

The chemical tracking system should be able to **import data from ChemiDraw software**.

📄 **Quick Table Recap – Software Quality Criteria**

Quality Criteria	What It Means	Easy Example
Modularity	Split into small parts/modules	Shopping app: login, cart, payment modules
Maintainability	Easy to fix bugs or make updates	Fix chemical report update in ≤ 20 hours
Testability	Easy to test if it's working properly	Keep code simple to reduce test effort
Flexibility	Easy to change or add features	Add print feature in 1 hour
Portability	Move software to different systems	Run same app on Windows & Linux
Reusability	Use same parts in other projects	Reuse chemical input feature in another app
Interoperability	Works with other software/systems	Import chemical data from ChemiDraw

📐 **What is the ISO-9126 Quality Framework?**

ISO-9126 is a **standard** created to define what makes software "good" or "high quality."
It gives us a **clear checklist** of the most important qualities that software should have.

📦 **It's divided into:**

- **6 Main Quality Characteristics**
- Each of these has **sub-parts** (called sub-characteristics), but in exams, the 6 main ones are usually enough.

🎯 **The 6 Quality Characteristics (with Examples)**

1️⃣ Functionality

📌 **What it means:**
Can the software do the **jobs it's supposed to do**?

💬 **Easy words:**
Does the software have the features people need?

🎯 **Example:**
An online banking app must let users transfer money, check balance, and pay bills.
If it can't do that, it's not functional — no matter how beautiful it looks.

2️⃣ Reliability

📌 **What it means:**
Can the software **keep working properly** without crashing?

💬 **Easy words:**
Does it work well **most of the time** and handle problems safely?

🎯 **Example:**
If a weather app crashes every 2 hours, it's unreliable.
A reliable app works even if the network is slow or data is missing.

3️⃣ Usability

📌 **What it means:**
Is the software **easy to learn and use**?

💬 **Easy words:**
Can a user use the software **without getting confused** or needing a manual?

🎯 **Example:**
A food delivery app with clear buttons and simple instructions is usable.
If it takes hours to figure out how to order — it's not usable.

4️⃣ Efficiency

📌 **What it means:**
Does the software use **less memory, time, and power** to do its job?

💬 **Easy words:**
Does it run fast and not waste computer resources?

🎯 **Example:**
If a photo app uses 90% of your phone's battery in 10 minutes, it's inefficient.
An efficient app loads quickly and uses little CPU/RAM.

5️⃣ Maintainability

📌 **What it means:**
Is it **easy to fix bugs or make changes** in the code?

💬 **Easy words:**
Can developers **understand, fix, or upgrade** the software easily?

🎯 **Example:**
If a program has clear code and is organized into small parts (modules), it's easy to maintain.
Otherwise, even a small bug might take days to fix.

6️⃣ Portability

📌 **What it means:**
Can the software be **moved and used** on different devices or systems?

💬 **Easy words:**
Can the same software run on **Windows, Linux, Mac, or different phones**?

🎯 **Example:**
A game that works on both Android and iPhone is portable.
If it only works on one phone model, it's not portable.

📄 **Quick Summary Table – ISO-9126 (For Revision)**

Quality Characteristic	Meaning	Easy Example
Functionality	Does the job as needed	Bank app lets you transfer money
Reliability	Works correctly and consistently	Weather app doesn't crash
Usability	Easy to use and learn	Food app with clear steps
Efficiency	Uses less time, memory, power	App loads fast and doesn't eat battery
Maintainability	Easy to fix or update	Code is organized and simple to change
Portability	Can run in different systems	App runs on both iOS and Android

📌 What are Quality Expectations?

When people talk about **software quality**, they are usually referring to what they **expect** from it. These expectations can come from two sides:

1. 👉 **External/Consumer Expectations** – What **users** want
2. 👉 **Internal/Producer Expectations** – What **developers** or **companies** care about

---

1️⃣ External / Consumer Expectations

These are the expectations from the **people who use the software** — regular users, customers, or clients.

✅ "Good enough" for the PRICE

They want software that **does the job well**, without needing it to be perfect or too expensive.

✳️ Two Key Terms:

- **Fit-for-use** = Doing the **right things**
  (The software should have the features the user needs.)

Example:
A student portal must show class routines, results, and notifications.
If it shows those, it's "fit-for-use."

- **Conformance** = Doing the **things right**
  (The software should work properly and not have bugs.)

Example:
A money transfer app should send the correct amount without errors.

📌 **Expectations vary depending on the type of software:**

Type of Software	What Consumers Expect	Example
General Applications	**Functionality** & **Reliability**	A text editor must save files correctly
GUI/Web-based Software	**Usability**	An e-commerce site should be easy to navigate
Safety-Critical Systems	**Safety**	Autopilot system must prevent airplane crashes

---

2️⃣ Internal / Producer Expectations

These are the expectations of the **developers**, **testers**, and **software companies**.

✅ **"Good enough" for the COST**

They want the software to be high quality but **within budget** and **on time**.

✳️ **Main Focus:**

- **Mirror the consumer side** → Developers also want the software to be functional and correct.
- They ensure this through **V&V** (Verification & Validation):
    - **Verification** = "Are we building the software right?"
    - **Validation** = "Are we building the right software?"

🔧 **Other Internal Expectations:**

Focus Area	Description	Example
Maintainability	Easy to fix or improve later	A program that allows quick updates to business rules
Interoperability	Able to **work with other systems**	A hospital system that connects with pharmacy databases
Modularity	System is divided into parts, helpful for **outsourcing** or future updates	Payment system is a separate module, easy to replace

---

🧠 **Summary Table**

Aspect	**Consumer Side**	**Producer Side**
Core Expectation	Good enough for **price**	Good enough for **cost**
Key Concepts	Fit-for-use, Conformance	V&V, Functionality, Correctness
Examples of Focus	Usability, Safety, Reliability	Maintainability, Interoperability, Modularity