Software Quality Assurance (SQA)

**Why Software Quality Matters**

**❗Deficiencies in software quality often result in:**

* **Costly emergency fixes**
  + Fixing software defects after deployment is much more expensive than catching them during development.
  + Example: A bug in a banking system may require an urgent fix overnight, involving high developer overtime costs and potential fines.
* **Damage to a brand’s reputation**
  + Poor quality software can harm the trust of users and customers.
  + Example: A buggy ride-sharing app can cause customers to shift to competitors.
* **Software company defaulters – large repayments**
  + Companies may have to offer refunds, pay penalties, or lose contracts if they ship defective products.

**Software Defects Are Extremely Costly In Terms Of:**

1. **Money**
   * Large financial losses. Example: Knight Capital lost $440 million due to a trading software bug in 2012.
2. **Loss of Life**
   * In safety-critical systems like medical equipment or defense, bugs can be fatal. Example: Therac-25 radiation overdoses caused deaths.
3. **Reputation**
   * Brand and customer loyalty can be destroyed, especially if errors are publicized.

**Famous Software Errors (Real Examples)**

|  |  |
| --- | --- |
| **System** | **Issue** |
| **Therac-25** | **Radiation overdose due to software bug** |
| **Patriot Missile System** | **Time drift caused failed interception** |
| **Mars Polar Lander (NASA)** | **Crash due to faulty signal interpretation** |
| **Ariane 5 (ESA)** | **Integer overflow explosion** |
| **2003 U.S. Blackout** | **Software alarm failure in monitoring system** |

**Standish Group Chaos Report**

|  |  |
| --- | --- |
| Project Status | Percentage |
| Succeeded | **35% (↑ from 16.2% in 1995)** |
| Failed | **19% (↓ from 31.1% in 1995)** |
| Challenged | **46% (↓ from 52.7% in 1995)** |

**What is Quality?**

**✳️ Experts define quality as:**

1. **Conformance to requirements**
   * Product meets **specified** requirements.
   * Example: If a spec requires the software to support 500 users, and it does, then it conforms.
2. **Fitness for use**
   * Product serves its **intended** purpose.
   * Example: A banking app should allow users to transfer funds securely—regardless of how it's implemented.

**User-Centric Definition of Quality**

1. **The system must do what it’s supposed to do** (correct functionality).
   * Called **validation**: Are we building the right product?
2. **The system must do it correctly** (correct implementation).
   * Called **verification**: Are we building the product right?

**What is Quality? (Expanded)**

Quality = Measurable characteristics such as:

* **Correctness** – Does the software produce the correct outputs?
* **Maintainability** – How easily can it be modified?
* **Portability** – Can it work on other platforms?
* **Testability** – How easily can it be tested?
* **Usability** – How user-friendly is it?
* **Reliability** – How often does it fail?
* **Efficiency** – Does it use resources well?
* **Integrity** – Can it resist unauthorized access?
* **Reusability** – Can code be reused in future projects?
* **Interoperability** – Can it work with other systems?

What is SQA? (Software Quality Assurance)

**✅ Definition:**

SQA is an umbrella activity applied **throughout the software process** to ensure software quality.

**⚠️ Why it’s needed:**

* Bugs are common in widely used software.
* Estimated loss in 2001: **$175 billion** worldwide due to bugs.

**🏢 Impacted Sectors:**

* **Banking systems**
* **Stock exchanges**
* **Medical institutions**
* **Educational institutions**
* **Social Security Administration**

**Importance of SQA**

**Most bugs can be avoided if:**

* A sound **software development process** is followed.
* There is strict **software quality control**.

**Prevention is Better Than Cure:**

SQA focuses on **preventing defects**, not just finding them.

**SQA vs. Software Testing**

|  |  |  |
| --- | --- | --- |
| Aspect | Software Testing | Software Quality Assurance |
| Goal | Detect bugs | Prevent bugs |
| Focus | Product correctness | Process correctness |
| Timing | After development stages | Throughout the process |
| Nature | Reactive | Proactive |
| Examples | Unit Testing, System Testing | Reviews, Audits, Standards |

* **Testing**: Running the software under controlled conditions to find bugs.
  + Example: Checking if a login form rejects wrong credentials.
* **SQA**: Ensures processes are followed so such bugs don’t occur in the first place.

**Elements of Software Quality Assurance**

|  |  |
| --- | --- |
| Element | Explanation |
| Standards | **ISO 9001, CMMI** |
| Reviews & Audits | **Code/design inspection** |
| Testing | **Functional, regression, etc.** |
| Error Analysis | **Root cause analysis** |
| Change Management | **Tracking changes properly** |
| Education | **Training for quality awareness** |
| Vendor Management | **Assessing quality of outsourced work** |
| Security & Safety | **Protecting against data loss or harm** |
| Risk Management | **Anticipating quality risks early** |

**SQA Tasks**

1. **Prepare an SQA Plan**
   * Tailored for each project.
2. **Define Software Process**
   * Ensure the process being followed is well-documented.
3. **Review Activities**
   * Verify compliance with the process.
4. **Audit Work Products**
   * Check if design/code/documents meet standards.
5. **Document Deviations**
   * Ensure all non-compliance is recorded and tracked.
6. **Report Noncompliance**
   * To senior management for corrective actions.

**SQA Plan Sections**

**1. Management Section**

* Describes SQA’s position in the org structure.

**2. Documentation Section**

* Lists all deliverables and documents.

**3. Standards, Practices, and Conventions Section**

* Lists standards (e.g., ISO), metrics to be used.

**4. Reviews and Audits Section**

* Schedule and methodology for reviews/audits.

**5. Test Section**

* References test plans and specifies how test data/results are recorded.

**6. Problem Reporting and Corrective Action**

* Defines procedures for tracking bugs/errors.

**7. Other**

* Tools, change control, training, risk management, record keeping.

**📊 Statistical SQA**

Focuses on **collecting, categorizing**, and **analyzing software defects**.

|  |  |
| --- | --- |
| **Step** | **Activity** |
| 1 | Collect data about errors |
| 2 | Categorize by type |
| 3 | Trace cause |
| 4 | Identify “vital few” causes |
| 5 | Fix root causes to reduce future defects |

We’ll continue from **Statistical SQA** onward in the next part. Would you like me to proceed now or take a pause here?

Let me know and I’ll continue with **Statistical SQA**, **Severity Levels**, **Reviews**, **Metrics**, and **Reliability** step-by-step in the same detailed style.

**👨‍👩‍👧‍👦 Peer Reviews**

Inspections to find defects early and improve quality

**🔍 Why Do Reviews?**

|  |  |
| --- | --- |
| **Benefit** | **Explanation** |
| Detect 80% of defects | If done properly |
| Detect logic/design flaws | Not just code bugs |
| Enforce coding standards | Uniformity and discipline |
| Encourage team learning | Knowledge sharing |

**✔️ Review Guidelines**

* **Review product, not the person**
* **Keep sessions < 30 mins**
* **Limit participants**
* **Follow a checklist**
* **Assign a coordinator**
* **Don’t debate solutions—log problems**
* **🔁 Defect Amplification and Removal**

|  |  |
| --- | --- |
| **Without Reviews** | **With Reviews** |
| Errors propagate between steps | Errors caught early |
| Cost increases with each step | Cost decreases |
| More rework later | Fewer bugs downstream |

**📌 Software Reliability (continued)**

**🔁 MTBF – Mean Time Between Failures**

MTBF stands for the **average time between two consecutive failures** in a system. It reflects the **reliability** of the software — higher MTBF means fewer failures and better reliability.

* **Formula:**

MTBF=MTTF+MTTR\text{MTBF} = \text{MTTF} + \text{MTTR}MTBF=MTTF+MTTR

Where:

* **MTTF (Mean Time To Failure)**: Average time the software runs before a failure occurs.
* **MTTR (Mean Time To Repair)**: Average time it takes to repair the software after a failure.

**🔁 Availability**

Availability is the **likelihood that the software system is operational** and meeting its intended purpose at any given moment.

* **Formula:**

Availability=MTTFMTTF+MTTR×100%\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \times 100\%Availability=MTTF+MTTRMTTF​×100%

**✅ Example:**

Suppose:

* MTTF = 100 hours
* MTTR = 10 hours

Then:

* MTBF = 100 + 10 = **110 hours**
* Availability = 100110×100=90.9%\frac{100}{110} \times 100 = 90.9\%110100​×100=90.9%

This means the software is operational **90.9% of the time**.

**📌 Why It Matters:**

* **Critical systems** like medical software, aircraft control, or banking systems need **high availability and reliability**.
* SQA teams aim to **minimize MTTR** and **maximize MTTF** through good design, testing, and process control.

**📊 Review Metrics and Their Use**

To **measure and improve the effectiveness** of reviews, we define **review metrics**:

**🔢 Defined Metrics:**

|  |  |
| --- | --- |
| **Metric** | **Description** |
| **Ep (Preparation Effort)** | Time spent preparing for the review |
| **Ea (Assessment Effort)** | Time spent during the review meeting |
| **Er (Rework Effort)** | Time to fix the issues identified |
| **WPS (Work Product Size)** | Size of the item reviewed (e.g., lines of code) |
| **Errminor** | Number of minor errors found |
| **Errmajor** | Number of major errors found |

**📐 Calculated Metrics:**

1. **Total Review Effort (Ereview):**

Ereview=Ep+Ea+Er\text{Ereview} = \text{Ep} + \text{Ea} + \text{Er}Ereview=Ep+Ea+Er

* + Shows the **total time invested** in reviews.

1. **Total Errors Found (Errtot):**

Errtot=Errminor+Errmajor\text{Errtot} = \text{Errminor} + \text{Errmajor}Errtot=Errminor+Errmajor

1. **Error Density:**

Error Density=ErrtotWPS\text{Error Density} = \frac{\text{Errtot}}{\text{WPS}}Error Density=WPSErrtot​

* + Measures **how many errors** exist per unit size of the work product.

**💸 Cost Effectiveness of Reviews**

We compare **review effort** vs. **testing effort** to determine how cost-effective reviews are:

**✅ Formula:**

Effort saved per error=Etesting−Ereviews\text{Effort saved per error} = \text{Etesting} - \text{Ereviews}Effort saved per error=Etesting−Ereviews

Where:

* **Etesting** = effort it would take to find & fix the error during testing
* **Ereviews** = effort to find & fix it early in reviews

**📌 Importance:**

* Reviews typically **catch errors earlier**, when they’re **cheaper to fix**.
* A $1 error in design can cost $1000 to fix after deployment.

**🔒 Software Reliability Metrics**

Reliability is further supported by collecting **operational failure data** over time.

**📌 Units of Measurement:**

* **Failures per hour/day/week**
* **Uptime vs. downtime**
* **Number of transactions before failure**

These help in:

* Predicting **system behavior**
* Planning **maintenance and updates**
* Measuring **progress towards reliability goals**

**✅ Summary of Key Reliability Metrics:**

| **Metric** | **Formula** | **Indicates** |
| --- | --- | --- |
| **MTTF** | Time to failure | Average reliability |
| **MTTR** | Time to repair | Maintainability |
| **MTBF** | MTTF + MTTR | Reliability |
| **Availability** | MTTFMTTF+MTTR×100%\frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \times 100\%MTTF+MTTRMTTF​×100% | Operational readiness |

**✅ What's Next?**

This completes the **SQA & Reviews** section of your syllabus. Up next is likely either:

* **Software Testing (Black-box & White-box)**
* or **Design Patterns**

Please confirm which topic you want to continue with next, and I’ll proceed in the **same structured and detailed format.**

**✅ Topic 1: Analyzing Review Metrics – Numerical Questions**

**Q1:**

A review was conducted on a module of 800 LOC.

* Preparation Effort = 3 hrs
* Assessment Effort = 2 hrs
* Rework Effort = 1.5 hrs
* Minor Errors = 12
* Major Errors = 4  
  📌 Find:  
  a) Total Review Effort  
  b) Total Errors  
  c) Error Density

**Q2:**

* Testing effort per error = 3 hrs
* Review effort per error = 1.2 hrs
* Number of errors = 10  
  📌 Find:  
  a) Total testing effort  
  b) Total review effort  
  c) Total effort saved

**Q3:**

A review team analyzed 1200 LOC:

* Ep = 4 hrs, Ea = 2 hrs, Er = 3 hrs
* Errors found: 5 major, 13 minor  
  📌 Find:  
  a) Total Review Effort  
  b) Total Errors  
  c) Error Density

**Q4:**

* 25 defects found during review
* 10 defects found during testing
* Review effort = 10 hrs  
  📌 Find:  
  a) Review Efficiency  
  b) Defect Removal Efficiency (DRE)

**Q5:**

A reviewer analyzed 900 LOC in 4 hours.  
📌 Find:  
a) Review rate (LOC/hour)  
b) Time to review 1800 LOC at same speed

**✅ Topic 2: Software Reliability – Numerical Questions**

**Q1:**

A system fails every 500 hours and takes 20 hours to repair.  
📌 Find:  
a) MTBF  
b) Availability

**Q2:**

* MTBF = 1000 hrs
* MTTR = 25 hrs  
  📌 Find:  
  a) MTTF  
  b) Availability

**Q3:**

A system fails 5 times in 10,000 hours.  
📌 Find:  
a) Failure rate  
b) MTTF

**Q4:**

* MTTF = 1200 hrs
* MTTR = 30 hrs  
  📌 Find:  
  a) MTBF  
  b) Availability  
  c) Unavailability

**Q5:**

* MTBF = 300 hrs
* MTTR = 6 hrs  
  📌 Find:  
  Expected uptime in 1 year (total hours = 8760 hrs)