**Q.2.a. The Requirements engineering process is accomplished through the execution of seven distinct functions. Specially, requirements engineering encompasses requirements eliciation, analysis, specification, verification, & management. Briefly describe thsese process.**

**Ans:**

**1. Requirements Elicitation**

**Definition:**  
This phase involves gathering requirements from stakeholders such as users, customers, and business analysts.

**Key Activities:**

* Conducting interviews, surveys, and workshops.
* Observing current workflows and systems.
* Identifying functional and non-functional requirements.

✅ **Example:**  
In a banking system, users might request **online fund transfers, balance inquiries, and transaction notifications**.

**2. Requirements Analysis**

**Definition:**  
Analyzing and refining collected requirements to check for completeness, consistency, and feasibility.

**Key Activities:**

* Identifying conflicting requirements.
* Evaluating technical feasibility.
* Prioritizing critical vs. optional requirements.

✅ **Example:**  
For an **e-commerce website**, ensuring that the **shopping cart, payment integration, and inventory management** work together without conflicts.

**3. Requirements Specification**

**Definition:**  
Documenting requirements clearly in a structured format, typically in an **SRS (Software Requirements Specification)** document.

**Key Activities:**

* Writing detailed descriptions of system behavior.
* Defining use cases, scenarios, and diagrams (UML, flowcharts).
* Categorizing requirements (must-have, optional, constraints).

✅ **Example:**  
A **hospital management system** may specify that patient records **must be encrypted** and accessible only by authorized personnel.

**4. Requirements Validation & Verification**

**Definition:**  
Ensuring that requirements are correct, complete, and meet stakeholder needs.

**Key Activities:**

* Conducting **reviews, inspections, and walkthroughs**.
* Checking against **business goals and constraints**.
* Ensuring requirements are **testable and measurable**.

✅ **Example:**  
In a **ride-hailing app**, validating that the **GPS tracking, fare calculation, and driver availability** work as intended.

**5. Requirements Documentation**

**Definition:**  
Maintaining structured records of all requirements to serve as a reference for developers, testers, and stakeholders.

**Key Activities:**

* Creating and updating the **SRS document**.
* Using tools like **JIRA, Confluence, or IBM Rational DOORS**.
* Ensuring documentation remains clear and up-to-date.

✅ **Example:**  
A **university portal system** should have well-documented **student registration, course selection, and grading policies**.

**6. Requirements Management**

**Definition:**  
Handling changes, tracking progress, and ensuring consistency throughout the software development lifecycle.

**Key Activities:**

* **Version control** to track requirement changes.
* Handling requirement updates due to market or business changes.
* Maintaining **traceability** between requirements and implementation.

✅ **Example:**  
A **mobile banking app** may need frequent updates to comply with **new security regulations** or add new features like **QR payments**.

**7. Requirements Negotiation**

**Definition:**  
Resolving conflicts between stakeholders and prioritizing key requirements based on constraints like **budget, time, and technical feasibility**.

**Key Activities:**

* Aligning business goals with technical limitations.
* Balancing stakeholder expectations.
* Making trade-offs between cost, time, and scope.

✅ **Example:**  
A **startup building a food delivery app** may have to choose between **AI-based recommendations** or **multi-city expansion** due to budget constraints.

**Q.2.b. differentiate between functional and non-functional requirementss. Provide examples of each and discuss the significance of non-functional requirements in the development process. Explain how non-functional requirements impact system architecture and design.**

**Difference Between Functional and Non-Functional Requirements**

According to **Sommerville ("Software Engineering")** and **Pressman ("Software Engineering: A Practitioner’s Approach")**, software requirements are divided into **functional** and **non-functional** requirements.

**Ans:**

**1. Comparison Table**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Functional Requirements** | **Non-Functional Requirements** |
| **Definition** | Describes **what** the system should do. | Describes **how** the system should perform. |
| **Focus** | System features, functionalities, and operations. | System performance, security, usability, and constraints. |
| **User Interaction** | Directly affects the user’s interaction with the system. | Affects the system’s behavior but is not directly visible to users. |
| **Measurement** | Can be tested by checking the system’s outputs. | Measured using performance metrics (e.g., speed, scalability, security). |
| **Examples** | - A user should be able to **log in**. - A bank app should **transfer money**. - An e-commerce site must allow **product search**. | - The system should **load within 3 seconds**. - Data should be **encrypted** for security. - The app must **handle 1 million users**. |
| **Documentation** | Included in **SRS (Software Requirements Specification)** as use cases and system features. | Documented as **quality attributes and system constraints**. |

**2. Significance of Non-Functional Requirements in Software Development**

Non-functional requirements ensure the system is **efficient, scalable, and secure**. Ignoring them can lead to serious issues such as **slow performance, security breaches, and system failures**.

🚨 **What happens if non-functional requirements are ignored?**  
❌ **Slow response time** → Users leave the system.  
❌ **Weak security** → Risk of hacking and data leaks.  
❌ **Low scalability** → The system crashes when too many users log in.

✅ **Example:**  
Imagine an **online food delivery app**:

* If it takes **too long to load (Performance)**, users will uninstall it.
* If hackers steal **customer payment details (Security)**, users will lose trust.
* If it **fails during peak hours (Scalability)**, the company loses revenue.

**3. Impact of Non-Functional Requirements on System Architecture and Design**

🔹 **Performance → Affects Server and Database Design**

* If the system must handle **millions of requests**, developers use **load balancing, caching, and optimized queries**.

🔹 **Security → Affects Authentication and Encryption**

* If the system must be **highly secure**, it requires **multi-factor authentication (MFA) and encryption protocols**.

🔹 **Scalability → Affects Cloud and Infrastructure Decisions**

* A growing system might need **cloud-based services like AWS or auto-scaling solutions**.

🔹 **Usability → Affects UI/UX Design**

* A user-friendly system needs **clear navigation, accessibility features, and responsive design**.

✅ **Example:**  
For a **global e-commerce website**,

* Performance requirements may lead to **CDN (Content Delivery Network) usage** to load pages faster.
* Security needs may demand **SSL encryption and fraud detection systems**.

**Q.2.c. Define People, Process and Product. Explain the process quality and product quality in details.**

**Ans:**

### ****People, Process, and Product in Software Engineering****

According to **Sommerville ("Software Engineering")** and **Pressman ("Software Engineering: A Practitioner’s Approach")**, software engineering focuses on three main factors: **People, Process, and Product**.

### ****1. People****

People are the most important part of software development. Without skilled people, software projects cannot be successful.

👨‍💻 **Who are involved?**

* **Software Engineers** – Write and test code.
* **Project Managers** – Plan and manage the development process.
* **Stakeholders (Clients, Users, Investors)** – Provide requirements and feedback.
* **Quality Assurance (QA) Team** – Ensure the product works correctly.

📌 **Example:** In a software development team, programmers write code, testers check for bugs, and managers ensure everything is on track.

### ****2. Process****

The process is the **set of steps** followed to develop software efficiently. A well-defined process ensures the project is completed **on time, within budget, and with high quality**.

🔹 **Types of Software Development Processes:**

* **Waterfall Model** – Step-by-step approach, good for simple projects.
* **Agile Model** – Flexible and fast, best for changing requirements.
* **Spiral Model** – Combines risk analysis and iterative development.

📌 **Example:** A company using the **Agile model** releases updates every 2 weeks, gathering feedback and making improvements quickly.

### ****3. Product****

The final software that is delivered to the customer is called the **product**. The product must meet customer needs and work efficiently.

✅ **Characteristics of a Good Software Product:**

* **Functionality** – It does what it is supposed to do.
* **Reliability** – It works without crashing.
* **Usability** – It is easy to use.
* **Performance** – It runs fast and smoothly.
* **Security** – It protects user data.

📌 **Example:** A mobile banking app must allow transactions (functionality), be secure from hackers (security), and work without slow loading times (performance).

## **Process Quality vs. Product Quality**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Process Quality** | **Product Quality** |
| **Definition** | How well the **software development process** is planned and executed. | How well the **final software product** meets user requirements. |
| **Focus** | Efficiency, organization, and risk management in development. | Usability, security, and performance of the final software. |
| **Measurement** | Evaluated using **CMMI (Capability Maturity Model Integration)** levels. | Evaluated using **customer feedback, bug reports, and performance tests**. |
| **Example** | Following Agile methodology to develop software faster. | A **fast, bug-free, and user-friendly** e-commerce website. |

### ****Importance of Process Quality****

* Reduces **errors and delays** in development.
* Ensures **smooth teamwork and better project management**.
* Saves **costs and time** by avoiding mistakes early.

✅ **Example:** A company using a **well-defined testing process** finds and fixes bugs early, reducing costs.

### ****Importance of Product Quality****

* Increases **customer satisfaction** and trust.
* Reduces **maintenance costs** and future bug fixes.
* Improves **business reputation and sales**.

✅ **Example:** A video streaming app with **high product quality** loads videos quickly and works on all devices.

**Q.2.d. What do you mean by Risk analysis and management? What step are involved in it. Explain it details.**

**Ans:**

**Risk analysis and management** is a process used in software engineering to identify, assess, and reduce risks that may affect a project’s success. Risks can lead to **delays, cost overruns, security issues, or failure of the software** if not managed properly.

## **Steps in Risk Analysis and Management**

### ****1. Risk Identification (Finding Possible Risks) 🔍****

In this step, the team identifies all potential risks that could affect the project. Risks can be related to:

* **Technology:** Using new or untested technology.
* **People:** Lack of skilled developers or team conflicts.
* **Cost & Budget:** Project running out of money.
* **Schedule:** Project delays due to unforeseen issues.
* **Security & Data:** Cyberattacks or data breaches.

🔹 Example: If a project depends on third-party software, a risk could be **delays in updates or discontinued support**.

### ****2. Risk Analysis & Assessment (Understanding the Impact) 📊****

Once risks are identified, they are analyzed to understand:

1. **Probability (How likely is the risk to happen?)**
2. **Impact (How much damage will it cause?)**

Risks are categorized as:  
✅ **High Risk:** Can cause serious failure (e.g., critical security flaws).  
✅ **Medium Risk:** Causes delays but can be fixed (e.g., staff shortage).  
✅ **Low Risk:** Minor inconvenience (e.g., UI design changes).

🔹 Example: If a team lacks experience with a required technology, it could lead to **delays and increased costs** (Medium-High risk).

### ****3. Risk Planning (Making a Risk Response Plan) 📝****

After analyzing risks, the team creates strategies to:

* **Avoid risks** (e.g., use stable technologies instead of new ones).
* **Mitigate risks** (e.g., provide team training to reduce skill gaps).
* **Transfer risks** (e.g., outsource risky parts of the project).
* **Accept risks** (e.g., if the risk has minimal impact, monitor it).

🔹 Example: If a project has a high security risk, the team may **invest in cybersecurity measures** to mitigate it.

### ****4. Risk Monitoring & Control (Checking Risks Regularly) 📈****

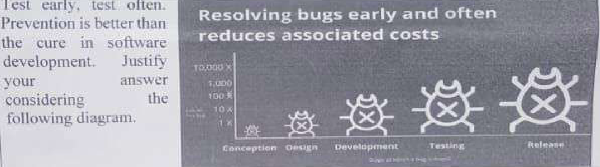
Since risks can change over time, the team must **continuously monitor** them throughout the software development life cycle (SDLC).

* Regular **meetings and reports** help track risks.
* If new risks appear, the plan is **updated** accordingly.

🔹 Example: If a risk of **developer attrition (staff leaving)** becomes real, the company must act fast by **hiring replacements** or **redistributing tasks**.

## **Why is Risk Management Important?**

✅ Helps avoid **major project failures**.  
✅ Reduces **unexpected delays** and **budget issues**.  
✅ Improves **software security and reliability**.  
✅ Increases **customer satisfaction** by delivering a stable product.

**Q.2.e. Test early, test often. Prevention is better than the cure in Software development. Justify your answer considering the following diagram.**

**Ans:**

### ****Justification: "Test Early, Test Often"****

The diagram supports the principle that **early testing reduces software development costs**. It highlights the idea that the **earlier a bug is found, the cheaper and easier it is to fix**.

#### **Key Points:**

1. **Bug Fixing Cost Increases Over Time**
   * If a defect is caught in the **conception or design phase**, it is **small and cheap** to fix.
   * If the defect is found **later in development, testing, or after release**, it becomes **much more expensive** and may require major rework.
2. **Prevention vs. Cure**
   * Early testing helps **prevent** serious issues rather than fixing them when they become **critical failures**.
   * Regular testing ensures a **smooth development process** and **high-quality software**.
3. **Improved Software Reliability**
   * Catching errors early **reduces failure risks**, ensuring a **better user experience** and a **stable product**.

### ****Conclusion:****

**"the cost of fixing defects grows exponentially as the software progresses."** The diagram visually proves that **early testing saves money, time, and effort**, reinforcing the need for **continuous testing throughout the development cycle**. 🚀

Another part of the above question:

Here are the answers to all your questions, based on **Sommerville** and **Pressman** books, in easy-to-understand English:

### ****1. Principles Followed During the Software Development Life Cycle (SDLC)****

The **Software Development Life Cycle (SDLC)** follows important principles to ensure quality and efficiency:

1. **Requirement Analysis:** Understand what the software needs to do.
2. **Planning:** Define project scope, timeline, and resources.
3. **Design:** Create system architecture and user interface.
4. **Development:** Write and test the code in small parts.
5. **Testing:** Find and fix errors before release.
6. **Deployment:** Make the software available to users.
7. **Maintenance:** Update and improve software after release.

These steps ensure the **software is high-quality, meets user needs, and works effectively**.

### ****2. Agile and Scrum in Software Engineering****

#### **Definition of Agile:**

Agile is a **flexible, iterative, and fast software development model** that helps teams adapt quickly to changes. It focuses on small, frequent updates rather than a single large release.

#### **Steps in Agile Model:**

1. **Requirement Gathering:** Discuss with customers and stakeholders.
2. **Design Requirements:** Plan system structure and features.
3. **Construction/Iteration:** Develop the software in small parts.
4. **Testing:** Perform regular testing and fix issues.
5. **Deployment:** Deliver working software to users.
6. **Feedback & Improvement:** Get user feedback and update the software.

#### **When Agile is Suitable:**

* When **requirements change frequently**.
* For **small to medium teams** working on evolving products.
* When **fast delivery and continuous improvements** are needed.

#### **Definition of Scrum:**

Scrum is a framework within Agile where work is divided into short, fixed periods called **sprints** (1–4 weeks). It includes roles like **Scrum Master, Product Owner, and Development Team**.

#### **Steps in Scrum:**

1. **Sprint Planning:** Decide what features to develop in the next sprint.
2. **Daily Stand-up Meetings:** Short meetings to discuss progress and challenges.
3. **Development:** Team builds and tests the features.
4. **Sprint Review:** Demonstrate the product to stakeholders.
5. **Sprint Retrospective:** Discuss what went well and what to improve.

#### **When Scrum is Suitable:**

* When teams work on **complex, evolving projects**.
* When fast delivery of features is needed.
* When customers need **frequent updates and improvements**.

### ****3. What is Software Engineering?****

**Software Engineering** is the **systematic approach** to developing software using engineering principles to ensure **quality, efficiency, and maintainability**.

#### **Applicability to Web Applications (WebApps):**

Yes, software engineering applies to **WebApps** but requires some modifications:

* **Scalability:** WebApps must handle many users at the same time.
* **Security:** WebApps need **data protection** and **secure login systems**.
* **Performance Optimization:** WebApps should load quickly.
* **Cross-Platform Compatibility:** WebApps must work on **different devices** and browsers.

### ****4. Requirements Engineering Process****

According to **Sommerville**, the **requirements engineering process** involves the following steps:

1. **Requirements Elicitation:** Gather requirements from users, customers, and stakeholders.
2. **Requirements Analysis:** Check for **conflicts, missing details, and feasibility**.
3. **Requirements Specification:** Document all requirements clearly in **Software Requirements Specification (SRS)**.
4. **Requirements Validation:** Review and confirm requirements with stakeholders.
5. **Requirements Management:** Track changes and keep requirements updated throughout development.

These steps ensure that **the software meets user needs and business goals**.

### ****5. Functional vs. Non-Functional Requirements****

| **Functional Requirements** | **Non-Functional Requirements** |
| --- | --- |
| Define **what the system does**. | Define **how the system performs**. |
| Example: **User login, payment processing**. | Example: **Speed, security, scalability**. |
| Focus on **features and functions**. | Focus on **performance, reliability, and user experience**. |
| Must be implemented **for software to work**. | Affects **system design and architecture**. |

#### **Importance of Non-Functional Requirements:**

* **Impact on System Architecture:** Determines **hardware, cloud servers, and security systems**.
* **User Experience:** Ensures fast response times and ease of use.
* **Compliance & Security:** Helps meet **industry standards and legal regulations**.

### ****6. People, Process, and Product****

#### **People:**

According to **Pressman**, people are the **most important** part of software development. **Software engineers, project managers, and testers** ensure quality development.

#### **Process Quality:**

A well-defined process ensures **efficiency, consistency, and lower risk** in software development. Following SDLC helps teams build **high-quality software**.

#### **Product Quality:**

A good software product must be **reliable, efficient, and user-friendly**. Testing and feedback help improve the **final product**.

### ****7. Risk Analysis and Management****

#### **Definition:**

Risk analysis and management is the process of **identifying, assessing, and reducing risks** in software projects.

#### **Steps in Risk Management:**

1. **Risk Identification:** Find potential risks (technical failures, security issues).
2. **Risk Analysis:** Evaluate **impact and likelihood**.
3. **Risk Prioritization:** Focus on **high-impact** risks first.
4. **Risk Mitigation:** Plan ways to **reduce or eliminate risks**.
5. **Risk Monitoring:** Continuously track risks throughout the project.

By managing risks properly, **projects can avoid failures and ensure software quality**.

### ****8. Justification: "Test Early, Test Often"****

#### **Why Early Testing is Important?**

The diagram shows that **the cost of fixing bugs increases at later stages** of development.

1. **Finding bugs early is cheaper:**
   * **During design:** Small changes are easy to fix.
   * **During development:** Fixing requires more effort.
   * **During testing or after release:** Fixing is **very expensive** and damages reputation.
2. **Prevention is better than fixing later:**
   * Early testing **reduces failures** and **improves software quality**.
   * Continuous testing ensures a **stable and error-free product**.

#### **Conclusion:**

Sommerville and Pressman emphasize that **early and frequent testing saves time, cost, and effort**. The diagram proves that **bugs should be detected and fixed as early as possible** to avoid **expensive rework and failures**.

These answers cover all your questions in **clear, simple language** while following **Sommerville and Pressman's** principles. Let me know if you need more details! 😊🚀