**1.Answer:**

**Anti-Lock Braking System in a Car:**

System Category: Safety-critical real-time system.

Approach: Waterfall or V-Model.

Rationale: Safety-critical systems such as anti-lock braking systems necessitate rigorous testing, verification, and validation. The structured approach of the Waterfall or V-Model places a strong emphasis on in-depth requirements analysis, design, and testing phases, ensuring a thorough and well-documented development process.

**Virtual Reality System for Software Maintenance:**

System Category: Innovative and intricate.

Approach: Agile or Iterative/Incremental.

Rationale: Projects involving innovation often require frequent iterations and the capacity to adapt. Agile or Iterative models are well-suited for such projects, allowing for continuous feedback, collaboration, and adjustments as the virtual reality system evolves.

**University Accounting System Replacement:**

System Category: Business-critical, focused on data.

Approach: Incremental or Spiral.

Rationale: When replacing an existing system, it's essential to migrate data and minimize disruption. Incremental or Spiral models enable a gradual development process, permitting the new system to evolve in stages while ensuring compatibility with the current system.

**Interactive Travel Planning System for Environmental Impact:**

System Category: User-centric, with a focus on sustainability.

Approach: Agile or Scrum.

Rationale: Systems centered around user experiences necessitate frequent user feedback and evolving requirements. Agile methodologies like Scrum shine in such scenarios by providing flexibility and avenues for continuous improvement, easily adapting to evolving environmental considerations and user needs.

**2.Answer**

* **Adaptation to Changing Requirements:** Incremental development offers the advantage of being adaptable to shifts in customer requirements as they evolve during the project's progression.
* **Gradual Specification Refinement:** Clients can gradually enhance and clarify their specifications as the software advances through iterative cycles, reducing the risk of misunderstandings.
* **Active Engagement and Feedback:** Ongoing customer involvement in each iteration creates opportunities for consistent feedback, ensuring that the software closely aligns with their expectations.
* **Early Delivery of Value:** Valuable functionality is delivered in small increments at an early stage, enabling customers to begin deriving benefits and make well-informed decisions regarding subsequent phases of development.
* **Fostering Trust and Collaboration:** Incremental development promotes the growth of trust and collaboration between the development team and the customer, resulting in a stronger partnership and an improved mutual understanding of system requirements over time.

**3.Answer:**Software Testing as an Incremental, Staged Process:

* **Early Detection of Flaws:** Incremental testing enables the timely identification of flaws, which ultimately reduces costs and the effort required for subsequent fixes.
* **Enhanced Bug Localization:** This approach simplifies the process of pinpointing the origins of defects, which in turn expedites the debugging process.
* **Improved Error Isolation:** Incremental testing effectively segregates errors within smaller code segments, streamlining the identification of root causes.
* **Strengthened Collaboration:** Incremental testing encourages collaboration between testers and developers, fostering a collective understanding of system behavior and requirements.
* **Continuous Validation:** Staged testing ensures that each development phase undergoes ongoing validation against established requirements, design specifications, and expected functionality.

**Are Developers the Best Fit for Testing Their Own Programs?**

* **Potential for Bias and Assumptions:** Developers, due to their familiarity with the code, may inadvertently overlook defects or make assumptions. Independent testers provide a fresh perspective, reducing the risk of bias.
* **Objective Assessment:** Independent testers offer an impartial evaluation from an end-user's perspective, free from the influence of the development process.
* **Specialized Testing Expertise:** Testers possess specialized skills, methodologies, and tools designed to create comprehensive test cases and execute systematic testing procedures.
* **Efficient Resource Allocation:** By allowing developers to focus on coding and development while delegating testing to dedicated testers, a more efficient development cycle ensues, resulting in a more thorough software evaluation.

**4.Answer:**

It is essential to clearly differentiate between developing user requirements and system requirements in the requirements engineering process for several key reasons:

**Alignment with user needs:** User requirements capture what users, stakeholders, and customers expect from the system. They focus on the "what" rather than the "how". Developing user requirements ensures that the system aligns with the actual needs and expectations of the people who will use it.

System requirements specify how the system will fulfill the user requirements in technical terms. They are more detailed and delve into the "how" of implementation. Separating user and system requirements help and ensure that technical details do not obscure the user's perspective, allowing for a clear focus on meeting user needs.

**Communication and understanding:** User requirements are typically expressed in natural language or user stories and are meant to be easily understandable by non-technical stakeholders. Developing them separately enhances communication between stakeholders and the development team, reducing the chances of misunderstandings.

System requirements are often technical and may include specifics about the architecture, databases, algorithms, and performance criteria. Keeping them separate helps technical teams understand how to implement the system while keeping user needs in mind.

**Flexibility and adaptability:** User requirements can change as user needs evolve or as new insights emerge during the project. By developing user requirements separately, it becomes easier to accommodate changes without impacting the detailed technical specifications of the system.

System requirements are less prone to frequent changes once they are well-defined. Separating them from user requirements allows for a stable technical foundation that can adapt to evolving user needs without constant modification.

**Validation and verification:** User requirements are validated by ensuring that they accurately reflect user expectations. Verification involves confirming that they have been correctly interpreted.

System requirements are verified by checking whether the system's technical components and features meet the specified criteria. Separating these two types of requirements allows for distinct validation and verification processes, reducing ambiguity.

**Traceability and accountability:** Traceability ensures that each user requirement can be linked back to a specific need or stakeholder. This helps maintain accountability for addressing user needs.

Traceability ensures that each system requirement can be traced back to one or more user requirements, demonstrating how they fulfill user needs. This traceability provides a clear understanding of the system's purpose.

**5th answer:**

Five Problems that Can Arise Without Effective Configuration Management

Some of the Configuration tools are:

* Ansible
* Chef
* Puppet
* Terraform

Without effective configuration management policies and processes, companies can face a variety of challenges. Here are five possible problems that may arise:

**Version control issues:** It can be difficult to keep track of different versions of software, documentation, and other assets without proper configuration management. This can lead to confusion about which version is the most recent, resulting in inconsistencies and errors in product development.

**Inaccurate documentation:** Documentation may become outdated or inconsistent if there is no strong configuration management process in place. This can make it difficult for team members to understand how different components interact or to troubleshoot issues effectively.

**Increased risk of errors and failures:** Without adequate configuration management, it is easy to accidentally introduce errors or bugs into software or systems. Changes made without proper oversight can lead to unforeseen problems, such as system failures or security vulnerabilities.

**Difficulty in auditing and compliance:** Companies often need to comply with industry standards, regulations, or internal quality assurance requirements. Ineffective configuration management makes it challenging to demonstrate compliance and pass audits, which could have legal or financial consequences.

**Wasted time and resources:** Inefficient configuration management processes can waste time and resources. Team members may spend excessive time searching for files or resolving issues caused by improper version control, delaying project timelines, and increasing costs.