Difference between plan-driven and agile:

In the aspect of planning phase, plan-driven process requires a very detailed plan of the stage, and the progress is measured against the plan. For agile method, planning is incremental, and it is easier to change for accommodating requirements

For documentation, plan-driven process produces a set of complete documents for the next phase, while agile method produces less complete documents to reduce the time wasted.

In the context of software testing, explain the importance of requirements traceability:

It defines the expectations of the testing team to help implementation and specification. Also, it ensures the team to build the product in the right way, and they are building the right product.

Importance of requirement management:

to keep track of individual requirements and maintain links between dependent requirements.

Traceability in the context of the software development lifecycle means an ability to ensure that:

1. All requirements are defined
2. Accepted requirements are broken down into development and test tasks, with references to each other
3. Source code is reviewed based on acceptance criteria during development
4. Changes at any time during the development lifecycle are traced, and collaboration is ensured
5. Test are performed and release is ready for deployment on time.

How users are involved when an incremental development:

Is to provide and prioritize new system requirements and to evaluate the iterations of the system.

What is software architecture:

A software architecture is a description of how a software system is organized.

Architectural design decisions:

* Is there a generic application architecture that can be used?
* How will the system be distributed?
* What architectural styles are appropriate?
* What approach will be used to structure the system?
* How will the system be decomposed into modules?
* What control strategy should be used?
* How will the architectural design be evaluated?
* How should the architecture be documented?

Difference between client-server and repository architecture?

In network requirement, it is crucial for client-server architecture but not for repository architecture.

For the distribution of the system, client-server design may require one or more servers to serve different clients. But while using repository design, one central repository is needed to integrate different sub-systems.

Black-box vs White-box testing:

|  |  |  |
| --- | --- | --- |
|  | Black box | White box |
| Testing subject | External behavior of the system | Internal functioning. |
| Test doer | Tester | Software developers. |
| Applied to | System Testing/Acceptance Testing. | Unit Testing/Integration Testing. |
| Time | Least time consuming. | Most time consuming. |
| Purpose | It is the behavior testing of the software. | It is the logic testing of the software. |
|  | It is also known as data-driven testing, functional testing, and closed box testing. | It is also known as clear box testing, code-based testing, structural testing, and transparent testing. |
| Algorithm testing | Black Box Test is not considered for algorithm testing. | White Box Test is well suitable for algorithm testing |

Key difference:

* Black Box Test only considers the system's external behavior, while White Box Test considers its internal functioning.
* Implementation knowledge is not required when applying Black Box Testing, unlike White Box Test.
* It takes less time to perform a Black Box Testing than a White Box Testing.

Key similarities:

White Box Testing and Black Box Testing have different functionalities. Still, both tests guarantee that it is possible to have the best version of the software and that the system performs without problems as desired.

Architecture and system characteristics

* Performance
  + Localize critical operations and minimize communications. Use large rather than fine-grain components.
* Security
  + Use a layered architecture with critical assets in the inner layers.
* Safety
  + Localize safety-critical features in a small number of sub-systems.
* Availability
  + Include redundant components and mechanisms for fault tolerance.
* Maintainability
  + Use fine-grain, replaceable components

Importance of architectural design:

It ensures the system is optimized for performance, security, safety, availability, and maintainability. Also, the system is more manageable with a good architecture, since every sub-system/ component has been clearly identified and linked.

Architectural patterns:

* Patterns are a means of representing, sharing, and reusing knowledge.
* An architectural pattern is a stylized description of good design practice, which has been tried and tested in different environments.
* Patterns should include information about when they are and when they are not useful.
* Patterns may be represented using tabular and graphical descriptions

Architectural patterns example:

* Model-view-controller(MVC) model
* The Layered architecture pattern
* Repository pattern
* Distributed system model(client-server architecture)
* Pipe and filter architecture