PSC IT Officer Questions Solution.

Solution by: Niranjan Kumar Das

Date : 7 March 2024
Kathmandu, Nepal

Technical Subject **Subject: SAD / Software Engineering**

Date: 2079/11/16 **Paper: Second** **Post: Computer Officer(Sangh)**

1. **Why user involvement throughout the systems developments of a project is necessary? Describe various systems analysis and design techniques in detail with example. (3+7=10)**

Ans: User involvement in the field of information system development is usually considered as vital mechanism to enhance system quality and ensure successful system implementation. The importance of user orientation in innovation activities is emphasized in business filed as well as in political and societal discussions. In today's competed and changing market situations, effective way to support market success are innovations originating from the needs of the customers. If users involve into the system development cycle, they can give more information details.

User involvement in information systems development efforts may instigated by assuming that such participation will offer valuable input to various technical decisions to be made. However, their participation may have a greater value because those decisions are more socio-technical than purely technical.

It has been found in numerous studies that effective involvement in system design produces the following benefits.

- Improved quality of the system arising from more accurate user requirements.
- Avoiding costly system features that the user did not want or cannot use.
- Improved levels of acceptance of the system.
- Greater understanding of the system by the user resulting in more effective use.
- Increased participation in decision-making in the organization.

2. **Explain about JAD and its purpose in system development process. Mention the roles of various JAD group members in the development of an information system. (5+5=10)**

Ans: Joint application development, frequently shortened to JAD, is a methodology that involves the client or end user in the design and development of a software application through a succession of collaborative workshops called JAD sessions. Its aim is to involve the customer (or end user) in defining the business need and developing a solution based on that need.

JAD is a unique approach to software and systems development because it involves clients, customers or end users throughout the product design and development lifecycle. It emphasizes a team-oriented development approach along with a group consensus-based problem-solving model.

The primary purpose of using JAD in the analysis phase is to collect systems requirements simultaneously from the key people involved with the system. Joint application design in project management is a process that helps in improving a project. Stakeholders and clients come together, and their different views are taken and considered. This helps in producing a quality project.

Key participants in joint application development.

Multiple parties are involved in JAD. They each play a key role in the JAD process, making contributions to ensure successful product development and delivery.

1. The ***executive sponsor*** is someone from a customer organization (that will ultimately use the product) who has the authority to make project-related decisions. They also set the vision for the project, resolve conflicts, communicate customer support and help minimize any resistance that the customer organization may express during the JAD process. The executive sponsor may be the customer's project manager, CIO, CISO, CEO, etc. They may attend JAD sessions but are not required to.
2. The ***JAD facilitator*** is a person from the provider's organization. They are mainly responsible for organizing and running the JAD sessions. They also organize, manage and execute various JAD activities; mediate disputes to minimize disagreements; and encourage user participation in the JAD sessions. Facilitators may be called upon to

summarize discussions to enable project decision-making by the executive sponsor. Facilitators usually do not contribute information to sessions. However, their role is critical in ensuring that the sessions proceed and result in useful inputs. The best JAD facilitators are trained in this role and have a good knowledge of various requirements gathering tools and techniques that will be used in the JAD sessions. They must also be impartial and not have any vested interest in the outcome of these sessions.

3. ***Users*** provide business expertise and are the main focus of JAD (and JAD sessions). They may provide strategic, tactical or operational direction for the project, and may represent various levels of the client organization. During JAD sessions, it's important to involve all the key user groups that are or may be affected by the project.
4. ***IT representatives*** provide technical inputs, giving technical advice and help development teams develop product visualizations, technical models and prototypes. They help translate users' view of the solution and business goals into implementable business requirements. They also clarify the project's technological constraints and support developers in using the available technology and tools. IT representatives may be the key developers from the product organization. They may also represent other functions of the company, such as data administration, business analysis, prototyping, production management or operations management.
5. ***Scribes*** are responsible for documenting JAD sessions and key points (e.g., action items). They may ask for clarifications and ensure that points are phrased correctly. The scribe and facilitator should never be the same person. Someone who plays both roles cannot play either role adequately or meet their individual responsibilities. They may also bog down the sessions and slow down the overall JAD process.
6. ***Observers*** observe JAD sessions and gather knowledge about user needs and session decisions. Their job is to watch and listen. They are not allowed to offer inputs or suggestions. Also, they can only interact with session participants during breaks or before and after sessions.

Advantages of Joint Application Development:

- These are some of the key benefits of JAD:
- Produce a design from the customer’s perspective.

- The teamwork between company and client helps to remove all risks.
- Due to the close interactions, progress is faster.
- JAD helps to accelerate design and also to enhance quality.
- JAD cheers the team to push each other which leads them to work faster and also to deliver on time.

Challenges faced in Joint Application Development:

- Sometimes opinions within the team members may differ which make difficult to align goals and maintain focus.
- On depending upon size of the project, in JAD people may have to spent significant amount of time.

Joint Application Development (JAD) may not be the answer what the organization needs, but it gives a much inclusive and flowing environment than others alike. JAD is used as a technique in early stages of a systems development of project for developing business system requirements. One of the purposes in JAD is to bring together MIS and end user in a structured workshop, which is accomplished by experienced JAD facilitators and customized, schedule agendas to assist the participant in completing high quality requirements. It is also seen that JAD process minimizes the development time, costs and errors.

Information Technology **Subject: SAD / Software Engineering**

Date: 2079/03/28 **Paper: I** **Post: Assistant IT Director (NRB)**

1. Why are all big companies looking for CISA as information System Auditor? (5)

Ans: The Certified Information Systems Auditor (CISA) is an industry certification in the field of audit, security and control of information systems.

We all think that the responsibility of a CISA professional is just auditing control, but no! the responsibilities of a Certified Information Systems Auditor also include:

- Measuring the evaluation of the IT control structure.
- Management and oversight of IT personnel.
- Evaluating resource management and the IT portfolio.
- Assessing the utility of the IT control framework.
- Evaluate IT procedures, standards, policies, and processes inside the organization.
- Assessing disaster recovery strategies and business continuity.

Doing all the duties that are mentioned is not an easy task. So recruiters search for professionals who are capable enough, and if you are a CISA professional, they will understand that you are well aware of the responsibilities.

2. What is the purpose of the System Development Life Cycle (SDLC)? (5)

Ans: PURPOSE OF THE SOFTWARE DEVELOPMENT LIFECYCLE (SDLC)

The purpose of SDLC is to ensure that software projects are completed on time, within budget, and meet the required quality standards.

The different phases of the SDLC and their importance are as follows;

PLANNING PHASE : It is a critical stage that sets the foundation for the rest of the project. The planning stage (which can also be referred to as feasibility stage) is exactly what it sounds like: the phase in which developers will plan for the upcoming project. It helps to define the problem and scope of any existing systems, as well as determine the

objectives and create a project plan that outlines how the project will be executed. And most importantly, Planning Phase sets the project schedule.

The importance of this phase are;

- It defined Project scope and objectives
- Conducts feasibility study.
- Creating a project plan
- Identifying project risks
- Establishing project governance

It sets the foundation for the rest of the SDLC process and helps to ensure that the project is aligned with the organization's goals and objectives.

ANALYSIS PHASE: The main objective of this phase is to gather and analyze requirements and create a functional specification that outlines how the software will function and what it will do. The analysis stage includes gathering all the specific details required for a new system as well as determining the first ideas for prototypes.

The importance of this phase are;

- Gathering requirements
- Analyzing requirements
- Creating a functional specification
- Developing use case

DESIGN PHASE: The design stage is a necessary precursor to the main developer stage. Developers will first outline the details for the overall application, alongside specific aspects, such as its User interfaces, System interfaces, Network and network requirements, Databases.

The main objective of this phase is to create a technical design and architecture for the software and develop a user interface design that is intuitive and easy to use.

The importance of this phase are as follows;

- Creating a technical design

- Developing a user interface design
- Creating prototypes
- Reviewing and refining the design

This is a critical phase to the success of the project. Design phase helps to ensure that the software is well-designed, meets the needs of the business and users, and is aligned with the organization's goals and objectives.

DEVELOPMENT PHASE: The development stage is the part where developers actually write code and build the application according to the earlier design documents and outlined specifications.

The development phase is critical to the success of the project. It is where the software is built and tested, and where the bulk of the project resources are expended. It is important to ensure that the software is well-built, tested, and integrated, and that it meets the requirements of the business and users.

The main importance of Development phase are;

- Writing code
- Conducting unit testing
- Integrating components
- Conducting system testing
- Refactoring code

TESTING PHASE: The code must be tested to make sure that there aren't any bugs and that the end-user experience will not negatively be affected at any point. During the testing stage, developers will go over their software with a fine-tooth comb, noting any bugs or defects that need to be tracked, fixed, and later retested. It's important that the software overall ends up meeting the quality standards that were previously defined in the SRS document.

The main objective of this phase is to perform unit, integration, system, and acceptance testing to ensure that the software meets the requirements and is free of defects. Importance of this phase are as follows;

- Unit testing
- Integration testing
- System testing
- Acceptance testing
- Bug fixing

DEPLOYMENT PHASE: The main objective of this phase is to install and deploy the software into the production environment and conduct user training to ensure that users can effectively use the software.

The importance of this phase are;

- Installing and configuring the software
- Conducting user training
- Deploying the software
- Monitoring and supporting the software

MAINTENANCE PHASE: After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care of by the developers.

The Maintenance phase is an important part of the SDLC process, and organizations should plan for ongoing support, updates, and enhancements to their software to ensure that it continues to meet the needs of users and the business.

3. List different types of system testing. Explain why we need software configuration management (SCM). (2+3=5)

Ans: System testing is also known as black-box testing because it focuses on the external parts of the system. It takes place after integration testing and before the acceptance testing. So this testing detects the issues within the integrated units of a system. In this way, it checks the design of the complete system and behavior per the end user's needs.

System Testing Process

- **Test Planning:** The test plan is a document that includes the information of your testing, like objectives, strategies, entry-exit criteria, software requirements, testing tools, guidelines, etc.
- **Test Case Design and Execution:** Create a test case for every feature with the test scenarios, description, process, and more. Then perform the tests and execute them.
- **Defect Tracking and Defect Management:** Track and record the defects with the defect tracking tools like- JIRA, Bugzilla, Trello, etc. Also, you must write about them for the developers to resolve them.
- **Reporting and Communication:** Create bug reports about defect reports to send to the developers. Try to follow an organizational method for this.

System Testing Types

A. Functional Testing

Unit Testing: This testing process checks whether each system unit or component works accordingly. This is the first testing step performed at the development stage, and it helps to fix the bugs before the development cycle.

Integration Testing: This testing aims to validate the different software units working together to achieve the expected functionality. Thus it establishes a steady communication between several components and subsystems of software.

Regression Testing: It would be best to modify your software every time as per your needs. So, regression testing ensures the system is working without default after very few changes are made, and the changes should not damage the system.

Teams will independently perform regression testing as part of your SDLC, but it mustn’t be done in silos.

- When legacy data is imported from an older system to a newer one, parallel tests are helpful to check if everything has been moved across seamlessly using an automated regression test suit.

- Find bugs by integrating BrowserStack with your CI tool and trigger regression testing across 3000+ desktop browsers and devices at every pull request

User Acceptance Testing: Generally, the end users perform this testing to verify the system meets the ultimate expectations. So, it goes through the users' perspective. You have to do this testing at the last stage of the SDLC.

B. Non-Functional Testing

Performance Testing: It checks the system's behavior instead of a certain workload. So it clarifies the system's speed, stability, responsiveness, memory usage, network usage, and more. It determines the system doesn't break when multiple users try to access it.

Security Testing: You have to examine the weak area of an application with security testing, which means you have to find out the part of the coding where there is a high chance of attack by a hacker. So it would be best to try and solve the issues through an attacker's mindset.

Usability Testing: Simply, usability testing checks the product's usability. So, it aims to test whether or not the system is easy to use. This test ensures the application has a 'user manual' or 'help menu' to simplify the product's usability to the end users.

Compatibility Testing: This testing examines the compatibility of a system with different browsers, platforms, operating systems, hardware, network, mobile devices, etc. So, it ensures the system can behave accurately with the variable models and devices.

- With BrowserStack, teams can access the latest Android and iOS devices and an exhaustive range of browsers – from legacy versions to the latest beta and dev releases.
- Remove the hassle of maintaining and upgrading an in-house device lab.
- One-click access to the latest/legacy versions of macOS and Windows.
- With built-in accessibility features like Screen Reader, Live lets you deliver a superior web experience to every user, including those with visual impairments.

In Software Engineering, Software Configuration Management(SCM) is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. The primary goal is to increase productivity with minimal mistakes. SCM is part of cross-disciplinary field of configuration management and it can accurately determine who made which revision.

The primary reasons for Implementing Technical Software Configuration Management System are:

- There are multiple people working on software which is continually updating.
- It may be a case where multiple version, branches, authors are involved in a software config project, and the team is geographically distributed and works concurrently.
- Changes in user requirement, policy, budget, schedule need to be accommodated.
- Software should able to run on various machines and Operating Systems.
- Helps to develop coordination among stakeholders.
- SCM process is also beneficial to control the costs involved in making changes to a system.

4. What is ERP? Explain its importance. (2+3=5)

Ans: Enterprise resource planning (ERP) is a platform companies use to manage and integrate the essential parts of their businesses. Many ERP software applications are critical to companies because they help them implement resource planning by integrating all the processes needed to run their companies with a single system.

An ERP software system can also integrate planning, purchasing inventory, sales, marketing, finance, human resources, and more.

KEY TAKEAWAYS

- ERP software can integrate all of the processes needed to run a company.
- ERP solutions have evolved over the years, and many are now typically web-based applications that users can access remotely.

- Some benefits of ERP include the free flow of communication between business areas, a single source of information, and accurate, real-time data reporting.
- There are hundreds of ERP applications a company can choose from, and most can be customized.
- An ERP system can be ineffective if a company doesn't implement it carefully.

Importance of ERP

- Save Money
- Improve Collaboration
- Offer Better Analytics
- Strengthen Productivity
- Enhance Customer Satisfaction
- Simplify Compliance and Gain Risk Management.
- Best Inventory Monitoring
- Amplify Production Planning & Resource Management

5. **Discuss the importance of Software Quality Assurance (SQA) in brief. What are the types of software maintenance? Give some design principles for maintability. (2+4+4)**

Ans: Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.

Software Quality Assurance is a process which works parallel to development of software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

Benefits of Software Quality Assurance (SQA):

- SQA produces high quality software.

- High quality application saves time and cost.
- SQA is beneficial for better reliability.
- SQA is beneficial in the condition of no maintenance for a long time.
- High quality commercial software increase market share of company.
- Improving the process of creating software.
- Improves the quality of the software.
- It cuts maintenance costs. Get the release right the first time, and your company can forget about it and move on to the next big thing. Release a product with chronic issues, and your business bogs down in a costly, time-consuming, never-ending cycle of repairs.

Software Maintenance refers to the process of modifying and updating a software system after it has been delivered to the customer. This can include fixing bugs, adding new features, improving performance, or updating the software to work with new hardware or software systems. The goal of software maintenance is to keep the software system working correctly, efficiently, and securely, and to ensure that it continues to meet the needs of the users.

Software maintenance is a continuous process that occurs throughout the entire life cycle of the software system. It is important to have a well-defined maintenance process in place, which includes testing and validation, version control, and communication with stakeholders.

Several Types of Software Maintenance

1. **Corrective Maintenance:** This involves fixing errors and bugs in the software system.
2. **Patching:** It is an emergency fix implemented mainly due to pressure from management. Patching is done for corrective maintenance but it gives rise to unforeseen future errors due to lack of proper impact analysis.
3. **Adaptive Maintenance:** This involves modifying the software system to adapt it to changes in the environment, such as changes in hardware or software, government policies, and business rules.

4. **Perfective Maintenance:** This involves improving functionality, performance, and reliability, and restructuring the software system to improve changeability.
5. **Preventive Maintenance:** This involves taking measures to prevent future problems, such as optimization, updating documentation, reviewing and testing the system, and implementing preventive measures such as backups.

Principles and practices to achieve maintainability in Software Design

Maintainability in software design refers to the ease with which a software system can be modified, extended, updated, or fixed over its entire lifecycle. A maintainable software design enables developers to make changes efficiently without introducing errors or compromising the system's stability.

significant majority of software development cost is maintenance than the initial development

Here are key principles and practices to achieve maintainability in software design.

Modularity and Separation of Concerns:

"Modularity" and "Separation of Concerns" are two important software design principles that contribute to creating well-structured, maintainable, and scalable software systems.

- Divide the software into smaller, cohesive modules that encapsulate specific functionalities.
- Maintain clear separation between different concerns, such as user interface, business logic, and data storage.

Single Responsibility Principle (SRP):

The **Single Responsibility Principle (SRP)** is one of the core principles of object-oriented programming and software design. It is part of the SOLID acronym, which represents a set of five design principles aimed at creating well-structured, maintainable, and scalable software systems.

- Each module, class, or component should have a single responsibility or reason to change. This minimizes the impact of changes on other parts of the system.

Encapsulation and Information Hiding:

Encapsulation and **Information Hiding** are two related concepts in object-oriented programming that contribute to creating well-structured and maintainable software systems.

- Hide implementation details and expose only essential interfaces, reducing dependencies and minimizing the effects of changes.

Loose Coupling and High Cohesion:

Loose Coupling and **High Cohesion** are two important design principles in software engineering that aim to create large-scale software projects, where managing complexity and facilitating collaboration among developers are critical for success.

- Design components that are loosely coupled (minimal dependencies) and highly cohesive (related functionality grouped together).
- Low coupling reduces the ripple effects of changes, while high cohesion enhances clarity and maintainability.

Clear and Readable Code:

- Write code that is easy to understand and read by following consistent naming conventions and meaningful variable/function names.
- Use comments and documentation to explain complex logic or algorithms.

Modular Testing:

Modular testing, also known as unit testing, is a software testing approach where individual components or units of a software application are tested in isolation. The goal of modular testing is to ensure that each unit of code works correctly on its own before integrating them into a larger system.

- Develop unit tests and integration tests for each module to catch defects early and ensure changes do not break existing functionality.

Version Control and Source Code Management:

Version Control (also known as **Source Code Management**) is a crucial aspect of software development that involves tracking and managing changes to source code and other project assets over time.

- Use version control systems (e.g., Git, Bitbucket, SVN) to track changes, collaborate with team members, and revert to previous versions if needed.

Documentation:

- Maintain up-to-date documentation that describes the architecture, design decisions, APIs, and usage instructions. This aids in understanding and maintaining the system.

Design Patterns:

It's important to note that design patterns are not strict rules; they're guidelines that can be adapted to specific contexts.

- Apply well-known design patterns to solve recurring design problems.
- Design patterns provide proven solutions that enhance maintainability.

Refactoring:

Refactoring is a disciplined technique in software development that involves restructuring existing code without changing its external behavior.

- Regularly review and refactor code to improve its structure, eliminate duplication, and simplify complex logic.
- Refactoring keeps the codebase clean and maintainable.

Avoid Over engineering:

Focus on delivering value while keeping the codebase clean and understandable. **Over engineering** refers to the practice of adding more complexity, features, or design elements to a software solution than what is actually necessary to meet the requirements.

- Strive for simplicity and avoid unnecessary complexity.
- Over engineering can lead to difficulties in understanding and maintaining the system.

Testing and **Automation:**

Testing and Automation are integral components of modern software development practices that contribute to building reliable, high-quality software systems.

- Implement automated testing for regression testing and continuous integration.
- Automated tests ensure that changes do not introduce new bugs.

Dependency Management:

Dependency Management is a critical aspect of software development that involves managing external dependencies, libraries, frameworks, and components that a software project relies on.

- Manage external dependencies carefully. Regularly update libraries and frameworks to benefit from bug fixes and improvements.

Adaptability and Future-Proofing:

Adaptable and future-proof software systems are more likely to thrive in dynamic environments and continue providing value to users and stakeholders over time.

- Anticipate future changes and design the system with flexibility in mind. Avoid hard-coding assumptions that might change over time.

By prioritizing maintainability in software design, developers can reduce the cost of ongoing development, minimize the risk of introducing defects, and extend the lifespan of the software. Well-maintained software is easier to evolve and adapt to changing requirements, technologies, and business needs.

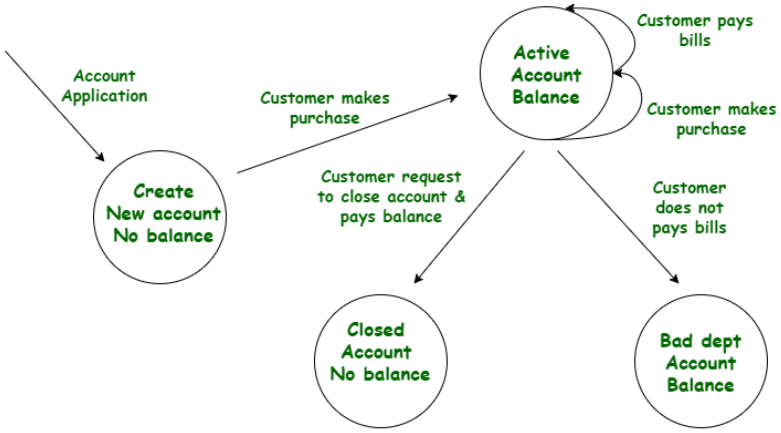
Maintainability can be achieved by adhering to standard design principles in software systems

6. **Compare object-oriented analysis and design with the structured analysis and design. Discuss different activities involved in each of the phase of the object-oriented development life cycle. (4+6=10)**

Ans: Analysis simple means to study or examine the structure of something, elements, and system requirements in detail, and methodical way. Structured analysis and Object-oriented analysis both are important for software development and are analysis techniques used in software engineering. But both are different from each other.

Structured Analysis and Object-Oriented Analysis (OOA) are two software development methodologies that are used to design and develop software systems. While they have some similarities, they differ in a number of key ways.

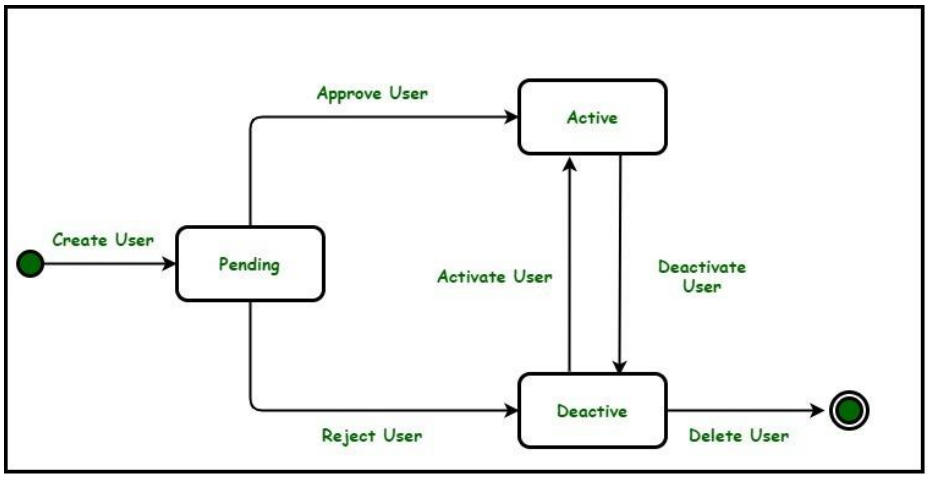
1. **Structured Analysis:** Structured analysis is a method of development that allows and gives permission to the analyst to understand and know about the system and all of its activities in a logical way. It is simply a graphic that is used to specify the presentation of the application. Example-



State Transition Diagram Example

2. **Object-Oriented Analysis:** Object-Oriented Analysis (OOA) is a technical approach generally used for analyzing and application designing, system designing, or even business designing just by applying object-oriented programming even with the use of visual modeling throughout the process of development to just simply guide the stakeholder communication and quality

of the product. it is actually a process of discovery where a team of developers understands and models all the requirements of the system.



Text

StateChart Diagram Example

Structured Analysis	**Object-Oriented Analysis**
The main focus is on the process and procedures of the system.	The main focus is on data structure and real-world objects that are important.
It uses System Development Life Cycle (SDLC) methodology for different purposes like planning, analyzing, designing, implementing, and supporting an information system.	It uses Incremental or Iterative methodology to refine and extend our design.
It is suitable for well-defined projects with stable user requirements.	It is suitable for large projects with changing user requirements.
Risk while using this analysis technique is high and reusability is also low.	Risk while using this analysis technique is low and reusability is also high.
Structuring requirements include DFDs (Data Flow Diagram), Structured Analysis, ER (Entity Relationship) diagram, CFD (Control Flow Diagram), Data Dictionary, Decision table/tree, and the State transition diagram.	Requirement engineering includes the Use case model (find Use cases, Flow of events, Activity Diagram), the Object model (find Classes and class relations, Object interaction, Object to ER mapping), Statechart Diagram, and deployment diagram.
This technique is old and is not preferred usually.	This technique is new and is mostly preferred.

Advantages of Structured Analysis:

- Clear and straightforward: Structured Analysis is a clear and straightforward methodology that is easy to understand and implement.
- Top-down approach: The top-down approach of Structured Analysis makes it easy to identify the high-level functions and processes involved in a software system, and to break them down into smaller, more manageable components.
- Emphasis on data: Structured Analysis places a strong emphasis on the data that a software system manipulates, making it easier to understand the relationships between data and processes.
- Well-suited for small to medium-sized systems: Structured Analysis is well-suited for small to medium-sized systems, where the focus is on breaking down complex systems into simpler, more manageable components.

Disadvantages of Structured Analysis:

- Limited scalability: Structured Analysis is limited in its scalability, and may become cumbersome when dealing with complex, large-scale systems.
- Lack of object orientation: Structured Analysis does not provide the object orientation and encapsulation benefits of OOA, making it more difficult to manage and maintain large systems over time.
- Limited ability to model complex relationships: Structured Analysis has a limited ability to model complex relationships between objects, making it less suitable for modeling large, complex systems.

Advantages of Object-Oriented Analysis (OOA):

- Reusable code: OOA enables the creation of reusable objects and design patterns, reducing the amount of code that needs to be written and improving the quality and consistency of the software.
- Scalability: OOA is more scalable than Structured Analysis, making it better suited for large, complex systems.

- Object orientation: OOA provides the benefits of object orientation, including encapsulation, inheritance, and polymorphism, making it easier to manage and maintain large systems over time.
- Better modeling of complex relationships: OOA enables better modeling of complex relationships between objects, making it better suited for modeling large, complex systems.

Disadvantages of Object-Oriented Analysis (OOA):

- Steep learning curve: OOA can be more difficult to understand and implement than Structured Analysis, especially for those who are not familiar with object-oriented programming.
- Bottom-up approach: The bottom-up approach of OOA can make it difficult to understand the high-level functions and processes involved in a software system, and to break them down into smaller, more manageable components.
- Emphasis on objects: OOA places a strong emphasis on objects, making it more difficult to understand the relationships between data and processes.
- Feeling lost in the vast world of System Design? It's time for a transformation! Enroll in our Mastering System Design From Low-Level to High-Level Solutions - Live Course and embark on an exhilarating journey to efficiently master system design concepts and techniques