- ## Maintenance : Software Maintenance,Software Supportability

Software maintenance is to give assurance that the software has the ability to satisfy the changing requirement of the user.

Needs of maintenance

- To correct defects found in product
- To improve design of software
- To improve performance of software product
- To add new features
- To improve communication with other software
- To transfer legacy software system into new software system

Types Software Maintenance

1. Corrective Maintenance

This maintenance contains changes and updates performed for fixing defects found in software products that are detected by the end user or tester.

This type is emergency maintenance in which unscheduled changes are done for temporarily keeping software in operation.

Because of pressure from management, the maintenance team released small codes for emergency corrections called patching.

20% of all maintenance activities are performed to do corrective maintenance.

2. Adaptive Maintenance

This maintenance involves making changes in software products to maintain the software product up-to-date.

Adaptive maintenance also is the modification of software to keep it usable after a change to its operating environment. Many factors can change an application's environment, including new technical knowledge, hardware and security threats.

It can also include applying changes to components of the software application which do not work properly due to change that is done in some other component of the software application.

3. Perfective Maintenance

Contains changes and updates performed to keep the software useful for a long duration.

These include new user requirements, new features for increasing the reliability and performance of software.

Perfective maintenance improves the software's functionality and usability.

Changes to the software's interface and user journey are part of perfective maintenance.

It is basically associated with including new or modified user requirements.

Perfective maintenance contributes 50% of total maintenance which is the largest of all the maintenance activities.

4. Preventive Maintenance

Contains changes and updates to avoid problems in software applications in future.

Objective is to solve the problems which are not major at this point but may cause serious issues in future.

This type of maintenance is also commonly known as future proofing. It includes making the software easier to scale more easily in response to increased demand and fixing latent faults before they become operational faults.

Preventive maintenance contributes 5% of all the maintenance activities.

Maintenance Log

A maintenance log is a document (often relatively simple) that records who did what, when, and why..

Maintenance logs are extremely useful for troubleshooting recurring or obscure problems, as they provide a record of all work performed on the system and may shed light on hard-to-spot interactions between seemingly unrelated symptoms.

Maintenance of your valued assets is an essential chore to perform at regular intervals. What would happen if you own a precious metal ornament and it gets rusty over time? It will certainly lose its value, therefore to keep your assets in a presentable and functional condition you need to maintain them on a timely basis. Same is the case with your property and vehicle that if they are not maintained with time they will be devalued.

To keep an organized maintenance program it is better to design a logo in which you can put your maintenance schedule of every equipment you own.

● Reengineering-,Business Process Reengineering

Software Re-engineering is a process of software development that is done to improve the maintainability of a software system. Re-engineering is the examination and alteration of a system to reconstitute it in a new form. This process encompasses a combination of sub-processes like reverse engineering, forward engineering, reconstructing, etc.

What is Re-engineering?

Re-engineering, also known as reverse engineering or software re-engineering, is the process of analyzing, designing, and modifying existing software systems to improve their quality, performance, and maintainability.

This can include updating the software to work with new hardware or software platforms, adding new features, or improving the software's overall design and architecture.

Software re-engineering, also known as software restructuring or software renovation, refers to the process of improving or upgrading existing software systems to improve their quality, maintainability, or functionality.

It involves reusing the existing software artifacts, such as code, design, and documentation, and transforming them to meet new or updated requirements.

Objective of Re-engineering

The primary goal of software re-engineering is to improve the quality and maintainability of the software system while minimizing the risks and costs associated with the redevelopment of the system from scratch. Software re-engineering can be initiated for various reasons, such as:

To describe a cost-effective option for system evolution.

To describe the activities involved in the software maintenance process.

To distinguish between software and data re-engineering and to explain the problems of data re-engineering.

The process of software re-engineering involves the following steps:

Software Re-engineering is a process of software development that is done to improve the maintainability of a software system. Re-engineering is the examination and alteration of a system to reconstitute it in a new form. This process encompasses a combination of sub-processes like reverse engineering, forward engineering, reconstructing, etc.

What is Re-engineering?

Re-engineering, also known as reverse engineering or software re-engineering, is the process of analyzing, designing, and modifying existing software systems to improve their quality, performance, and maintainability.

This can include updating the software to work with new hardware or software platforms, adding new features, or improving the software's overall design and architecture.

Software re-engineering, also known as software restructuring or software renovation, refers to the process of improving or upgrading existing software systems to improve their quality, maintainability, or functionality.

It involves reusing the existing software artifacts, such as code, design, and documentation, and transforming them to meet new or updated requirements.

Objective of Re-engineering

The primary goal of software re-engineering is to improve the quality and maintainability of the software system while minimizing the risks and costs associated with the redevelopment of the system from scratch. Software re-engineering can be initiated for various reasons, such as:

To describe a cost-effective option for system evolution.

To describe the activities involved in the software maintenance process.

To distinguish between software and data re-engineering and to explain the problems of data re-engineering.

The process of software re-engineering involves the following steps:

- Planning: The first step is to plan the re-engineering process, which involves identifying the reasons for re-engineering, defining the scope, and establishing the goals and objectives of the process.
- Analysis: The next step is to analyze the existing system, including the code, documentation, and other artifacts. This involves identifying the system's strengths and weaknesses, as well as any issues that need to be addressed.

- Design: Based on the analysis, the next step is to design the new or updated software system. This involves identifying the changes that need to be made and developing a plan to implement them.
- Implementation: The next step is to implement the changes by modifying the existing code, adding new features, and updating the documentation and other artifacts.
- Testing: Once the changes have been implemented, the software system needs to be tested to ensure that it meets the new requirements and specifications.
- Deployment: The final step is to deploy the re-engineered software system and make it available to end-users.

Re-engineering can be done for a variety of reasons, such as:

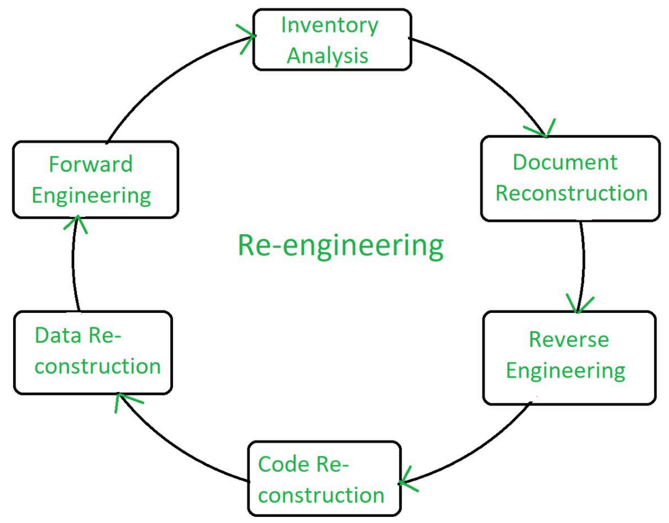
- To improve the software's performance and scalability: By analyzing the existing code and identifying bottlenecks, re-engineering can be used to improve the software's performance and scalability.
- To add new features: Re-engineering can be used to add new features or functionality to existing software.
- To support new platforms: Re-engineering can be used to update existing software to work with new hardware or software platforms.
- To improve maintainability: Re-engineering can be used to improve the software's overall design and architecture, making it easier to maintain and update over time.
- To meet new regulations and compliance: Re-engineering can be done to ensure that the software is compliant with new regulations and standards.
- Improving software quality: Re-engineering can help improve the quality of software by eliminating defects, improving performance, and enhancing reliability and maintainability.
- Updating technology: Re-engineering can help modernize the software system by updating the technology used to develop, test, and deploy the system.
- Enhancing functionality: Re-engineering can help enhance the functionality of the software system by adding new features or improving existing ones.
- Resolving issues: Re-engineering can help resolve issues related to scalability, security, or compatibility with other systems.

The process of software re-engineering typically involves the following steps:

- Assessment: The existing software system is analyzed to identify its strengths and weaknesses, and to determine the extent of the required reengineering effort.
- Planning: A plan is developed to guide the re-engineering effort, including the identification of goals, objectives, and performance metrics.
- Requirements analysis: The requirements of the new system are defined based on user needs and business requirements.
- Architecture and design: The architecture and design of the new system are developed, taking into account the requirements of the new system and the limitations of the existing system.
- Implementation: The new system is developed and implemented, either by modifying the existing system or by developing a new system from scratch.
- Testing: The new system is tested to ensure that it meets the requirements of the new system and is free from errors and defects.
- Maintenance: The new system is maintained over time to ensure that it remains reliable, performant, and secure.

Steps involved in Re-engineering

- Inventory Analysis
- Document Reconstruction
- Reverse Engineering
- Code Reconstruction
- Data Reconstruction
- Forward Engineering



Re-engineering Cost Factors

1. The quality of the software to be re-engineered.
2. The tool support is available for re-engineering.
3. The extent of the required data conversion.
4. The availability of expert staff for re-engineering.

Factors Affecting Cost of Re-engineering

Re-engineering can be a costly process, and there are several factors that can affect the cost of re-engineering a software system:

1. Size and complexity of the software: The larger and more complex the software system, the more time and resources will be required to analyze, design, and modify it.
2. Number of features to be added or modified: The more features that need to be added or modified, the more time and resources will be required.
3. Tools and technologies used: The cost of re-engineering can be affected by the tools and technologies used, such as the cost of software development tools and the cost of hardware and infrastructure.
4. Availability of documentation: If the documentation of the existing system is not available or is not accurate, then it will take more time and resources to understand the system.

5. Team size and skill level: The size and skill level of the development team can also affect the cost of re-engineering. A larger and more experienced team may be able to complete the project faster and with fewer resources.

6. Location and rate of the team: The location and rate of the development team can also affect the cost of re-engineering. Hiring a team in a lower-cost location or with lower rates can help to reduce the cost of re-engineering.

7. Testing and quality assurance: Testing and quality assurance are important aspects of re-engineering, and they can add significant costs to the project.

8. Post-deployment maintenance: The cost of post-deployment maintenance such as bug fixing, security updates, and feature additions can also play a role in the cost of re-engineering.

Advantages of Re-engineering

1. Reduced Risk: As the software is already existing, the risk is less as compared to new software development. Development problems, staffing problems and specification problems are the lots of problems that may arise in new software development.

2. Reduced Cost: The cost of re-engineering is less than the costs of developing new software.

3. Revelation of Business Rules: As a system is re-engineered , business rules that are embedded in the system are rediscovered.

4. Better use of Existing Staff: Existing staff expertise can be maintained and extended to accommodate new skills during re-engineering.

5. Improved efficiency: By analyzing and redesigning processes, re-engineering can lead to significant improvements in productivity, speed, and cost-effectiveness.

6. Increased flexibility: Re-engineering can make systems more adaptable to changing business needs and market conditions.

7. Better customer service: By redesigning processes to focus on customer needs, re-engineering can lead to improved customer satisfaction and loyalty.

8. Increased competitiveness: Re-engineering can help organizations become more competitive by improving efficiency, flexibility, and customer service.

9. Improved quality: Re-engineering can lead to better quality products and services by identifying and eliminating defects and inefficiencies in processes.

10. Increased innovation: Re-engineering can lead to new and innovative ways of doing things, helping organizations to stay ahead of their competitors.
11. Improved compliance: Re-engineering can help organizations to comply with industry standards and regulations by identifying and addressing areas of non-compliance.

Disadvantages of Re-engineering

Major architectural changes or radical reorganizing of the systems data management has to be done manually. Re-engineered system is not likely to be as maintainable as a new system developed using modern software Re-engineering methods.

1. High costs: Re-engineering can be a costly process, requiring significant investments in time, resources, and technology.
2. Disruption to business operations: Re-engineering can disrupt normal business operations and cause inconvenience to customers, employees and other stakeholders.
3. Resistance to change: Re-engineering can encounter resistance from employees who may be resistant to change and uncomfortable with new processes and technologies.
4. Risk of failure: Re-engineering projects can fail if they are not planned and executed properly, resulting in wasted resources and lost opportunities.
5. Lack of employee involvement: Re-engineering projects that are not properly communicated and involve employees, may lead to lack of employee engagement and ownership resulting in failure of the project.
6. Difficulty in measuring success: Re-engineering can be difficult to measure in terms of success, making it difficult to justify the cost and effort involved.
7. Difficulty in maintaining continuity: Re-engineering can lead to significant changes in processes and systems, making it difficult to maintain continuity and consistency in the organization.

- ## Software Reengineering

  When we are required to update the software application without affecting its functionality so that it can stay in the current market, it is known as software re-engineering.

  Sometimes developers observe that some components of software products require more maintenance than other components and such components require re-engineering.

The re-Engineering procedure requires the following steps:

Decide which components of the software we want to re-engineer. Is it the complete software or just some components of the software?

Do Reverse Engineering to learn about existing software functionalities.

Perform restructuring of source code if needed for example modifying functional-Oriented programs in Object-Oriented programs

Perform restructuring of data if required

Use Forward Engineering ideas to generate re-engineered software.

The Software Reengineering process basically undergoes following phases.

(1) Reverse engineering, (2) Restructuring, (3) Forward engineering (4) Component reusability

1. Reverse Engineering

It is a procedure to get system specification by analyzing and understanding the existing system.

This procedure can reverse software development life cycle models. ie. Go from the maintenance phase to requirement gathering phase.

2. Restructuring of source code

Restructuring of source code is a procedure of performing the restructuring and re-construction of the already existing software.

It is associated with re-arranging the source code. In Restructuring of source code, we can perform source code-restructuring or data-restructuring or both.

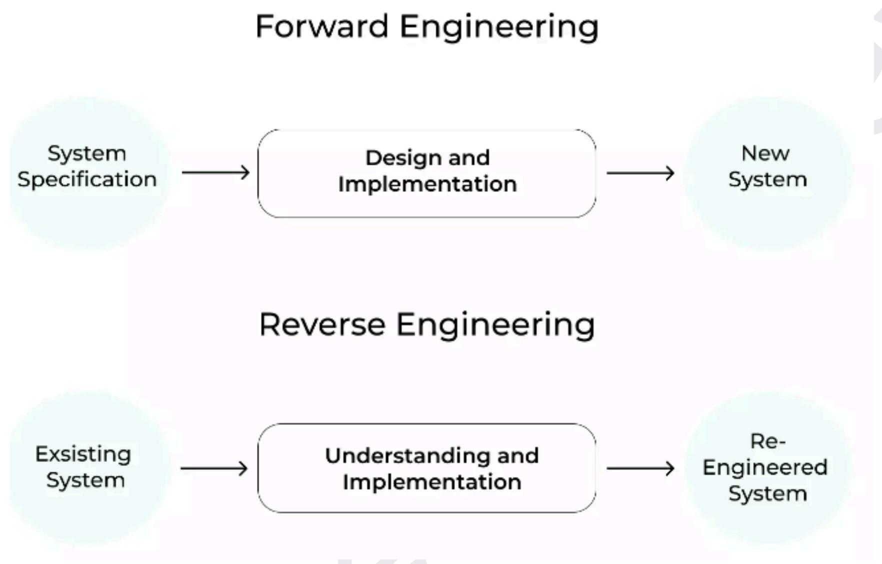
Restructuring does not affect the functionality of the existing software. It improves reliability and maintainability of software.

Program parts due to which errors occur frequently can be changed or updated in restructuring.

3. Forward Engineering

Forward engineering is a procedure of finding desired software product from the requirements in hand which is the output of reverse engineering.

Forward engineering is similar to software engineering procedure with the difference that it is always performed after reverse engineering.



4. Component Reusability

A component is an element of source code which performs an independent task.

Software component reuse is the software engineering practice of creating new software applications from existing components, rather than designing and building them from scratch.