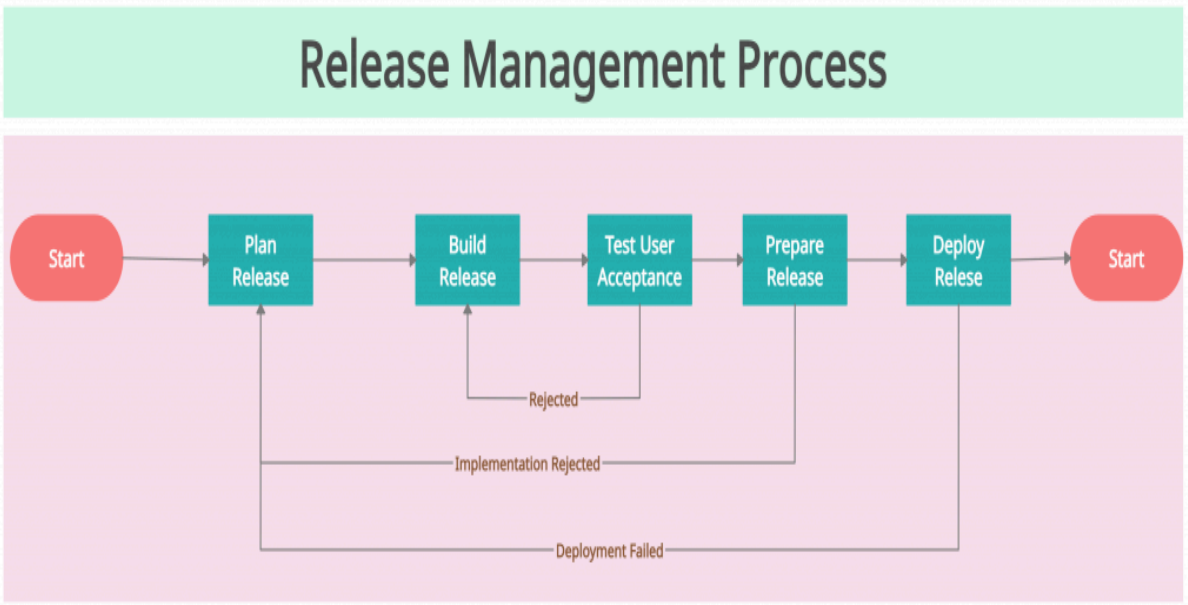
UNIT -5

# #Project Release Management-

- Product Release Management is the process of planning, scheduling, coordinating, and delivering a product (or software update) to the customers or users.
- It ensures that the right version of the product is released at the right time, with quality assurance, proper documentation, and minimal risks.
- System release managers are responsible for deciding when system can be released to customers, managing process of creating release and distribution media and documenting release for only ensurity that it can be re-created same as distributed if it is possible




**Key Stages in Product Release Management**

**1. Planning**

- Identify the features or updates to be released.
- Set the timeline and release date.

**2. Development**

- Developers start building the feature or product.
- Code is written, reviewed, and integrated.

**3. Testing**

- Perform different types of testing:
  - **Unit Testing** – testing individual components
  - **Integration Testing** – testing how parts work together
  - **User Acceptance Testing (UAT)** – testing from the user's perspective

**4. Release Preparation**

- Finalize the version number (e.g., v2.1).
- Prepare **release notes**, **documentation**, and **user manuals**.
- Train customer support teams.

**5. Deployment**

- Release the product to users.
- Can be done in **phases** (e.g., 10% of users first) or **globally**.
- Monitor for any immediate issues or bugs.

**Example**: The app update is pushed to app stores (Google Play, App Store).

**6. Post-Release Monitoring**

- Monitor system performance and user feedback.
- Fix bugs (if any) quickly.
- Collect feedback for the next version.

# # RISK MANAGEMENT

- Risk Management is the process of identifying, analyzing, planning for, monitoring, and controlling risks throughout a project or system lifecycle.
- The main goal is to minimize negative impact on the project’s objectives—cost, time, quality, and scope.

- Risk Management is a systematic process of recognizing, evaluating, and handling threats or risks that have an effect on the finances, capital, and overall operations of an organization. These risks can come from different areas, such as financial instability, legal issues, errors in strategic planning, accidents, and natural disasters.
- The main goal of risk management is to predict possible risks and find solutions to deal with them successfully.

### Difference between Reactive and proactive

| Basis | Proactive Risk Management | Reactive Risk Management |
|---|---|---|
| **Methods** | Identifies and mitigates problems before they arise. | Focuses on hazards after they arise. |
| **Duration** | Long-term outlook. | Temporal viewpoint. |
| **Concentrate** | Avoidance and reduction of damage. | Reaction and recuperation. |
| **Being proactive** | Reacts proactively to dangers. | Actively detects and controls hazards. |
| **Efficiency** | Lessens the possibility and effect of hazards. | Deals with hazards after they have already happened. |
| **Price** | It may need an initial outlay of funds, but it can ultimately result in cost savings. | It may cost more because of unforeseen circumstances. |
| **Planning** | Places a focus on risk assessment and planning. | Not enough thorough planning or preparation. |

# SOFTWARE RISK:

- Software risk analysis in software development is a systematic process that involves identifying and evaluating any problem that might happen during the creation, implementation, and maintaining of software systems.
- It can guarantee that projects are finished on schedule, within budget, and with the appropriate quality. It is a crucial component of software development.

### Possible Scenarios of Risk Occurrence

**1.Unknown Unknowns**

These risks are unknown to the organization and are generally technology related risk due to this these risks are not anticipated. Organizations might face unexpected challenges, delays, or failures due to these unexpected risks. Lack of experience with a particular tool or technology can lead to difficulties in implementation.

**Example**

Suppose an organization is using cloud service from third-party vendors, due to some issues third party vendor unable to provide its service. In this situation organization have to face an unexpected delay.

**2.Known Knowns**

These are risks that are well-understood and documented by the team. Since these risks are identified early, teams can plan for mitigation strategies. The impact of known knowns is usually more manageable compared to unknown risks.

**Example**

The shortage of developers is a known risk that can cause delays in software development.

**3.Known Unknowns**

In this case, the organization is aware of potential risks, but the certainty of their occurrence is uncertain. Organization should get ready to deal with these risks if they happen. Ways to deal with them might include making communication better, making sure everyone understands what's needed, or creating guidelines for how to manage possible misunderstandings.

**Example**

The team may be aware of the risk of miscommunication with the client, but whether it will actually happen is unknown.

## Types of software risks:

| TYPE | EXAMPLE |
|---|---|
| **Project Risks** | Team leaves mid-project |
| **Technical Risks** | Software doesn’t work on all devices |
| **Business Risks** | Product doesn’t match market needs |
| **Security Risks** | Hackers can attack the system |

# RISK IDENTIFICATION:

- Identifying risk is one of most important or essential and initial steps in risk management process. By chance, if failure occurs in identifying any specific or particular risk, then all other steps that are involved in risk management will not be implemented for that particular risk.
- For identifying risk, project team should review scope of program, estimate cost, schedule, technical maturity, parameters of key performance, etc. To manage risk, project team or organization are needed to know about what risks it faces, and then to evaluate them.
- Generally, identification of risk is an iterative process. It basically includes generating or creating comprehensive list of threats and opportunities that are based on events that can enhance, prevent, degrade, accelerate, or might delay successful achievement of objectives. In simple words, if you don't find or identify risk, you won't be able to manage it.

**There are many different types of risks which affects the software project:**

**1.** Technology risks

2. Tools risks

3. Estimation risks

4. People risks

5. Requirement risks

6. Organizational risks

**Methods for Identifying Risks:**

**1. Checklist Analysis –**

- Checklist Analysis is type of technique generally used to identify or find risks and manage it.
- The checklist is basically developed by listing items, steps, or even tasks and is then further analyzed against criteria to just identify and determine if procedure is completed correctly or not. It is list of risk that is just found to occur regularly in development of software project.

**2. Brainstorming –** This technique provides and gives free and open approach that usually encourages each and every one on project team to participate.

- It also results in greater sense of ownership of project risk, and team generally committed to managing risk for given time period of project. It is creative and unique technique to gather risks spontaneously by team members. The team members identify and determine risks in 'no wrong answer' environment.
- This technique also provides opportunity for team members to always develop on each other's ideas.

**3. Casual Mapping –** Causal mapping is method that builds or develops on reflection and review of failure factors in cause and effect of the diagrams. It is very useful for facilitating learning with an organization or system simply as method of project-post evaluation. It is also key tool for risk assessment.

**4. SWOT Analysis –** Strengths-Weaknesses-Opportunities-Threat (SWOT) is very technique and helpful for identifying risks within greater organization context. It is generally used as planning tool for analyzing business, its resources, and also its environment simply by looking at internal strengths and weaknesses and opportunities and threats in external environment.

**5. Flowchart Method –** This method allows for dynamic process to be diagrammatically represented in paper. This method is generally used to represent activities of process graphically and sequentially to simply identify the risk.

# RISK PROJECTION

- Risk projection is the process of **predicting**:
- How likely a risk is to happen (**probability**)
- How much damage it could cause (**impact**)
- It helps the software team to **prioritize** the most serious risks and plan actions to handle them.

## Steps of Risk Projection:

1. **Identify Risks**
   → List possible problems that may occur during the project.
2. **Estimate Probability**
   → Decide how likely the risk is (e.g., low, medium, high or in %).
3. **Estimate Impact**
   → Estimate how serious the effect of the risk will be if it occurs.
4. **Calculate Risk Exposure**
   → Use the formula:
   **Risk Exposure (RE) = Probability × Impact**

### Why Risk Projection is Important:

- Helps in **planning ahead**
- Saves **time and cost**
- Prevents **project failure**
- Helps in **making decisions** and assigning responsibilities

## RISK REFINEMENT

- Risk refinement means **breaking a big risk** into **smaller, more specific sub-risks**, so we can understand the causes and take proper action to prevent them.
- It gives **more detailed information** about each risk and how to deal with it.

### Steps of Risk Refinement:

1. **Choose a major risk** (for example: "System crash")
2. **Break it into sub-risks** (e.g., "memory leak", "CPU overload")
3. **Find the causes** of each sub-risk
4. **Create a solution plan** for each

## # RMMM:

- A risk management technique is usually seen in the software Project plan. This can be divided into Risk Mitigation, Monitoring, and Management Plan (RMMM).
- In this plan, all works are done as part of risk analysis. As part of the overall project plan project manager generally uses this RMMM plan.

## Risk Mitigation: (Risk Avoidance)

It is an activity used to avoid problems.

Steps for mitigating the risks as follows.

1. Finding out the risk.
2. Removing causes that are the reason for risk creation.
3. Controlling the corresponding documents from time to time.
4. Conducting timely reviews to speed up the work

## Risk Monitoring: (Keeping an eye)

It is an activity used for project tracking.
It has the following primary objectives as follows.

1. To check if predicted risks occur or not.
2. To ensure proper application of risk aversion steps defined for risk.
3. To collect data for future risk analysis.
4. To allocate what problems are caused by which risks throughout the project.

## Risk Management:(Taking Action)

- It assumes that the mitigation activity failed and the risk is a reality. This task is done by Project manager when risk becomes reality and causes severe problems.
- If the project manager effectively uses project mitigation to remove risks successfully then it is easier to manage the risks. This shows that the response that will be taken for each risk by a manager. The main objective of the risk management plan is the risk register. This risk register describes and focuses on the predicted threats to a software project.

# #SOFTWARE MAINTENANCE

- Software Maintenance refers to the process of modifying and updating a software system after it has been delivered to the customer. This involves fixing bugs, adding new features, and adapting to new hardware or software environments.
- Effective maintenance is crucial for extending the software's lifespan and aligning it with evolving user needs. It is an essential part of the software development life cycle (SDLC), involving planned and unplanned activities to keep the system reliable and up-to-date.

- 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.
- It's important to note that software maintenance can be costly and complex, especially for large and complex systems. Therefore, the cost and effort of maintenance should be taken into account during the planning and development phases of a software project.

## Several Key Aspects of Software Maintenance

1. **Bug Fixing:** The process of finding and fixing errors and problems in the software.
2. **Enhancements:** The process of adding new features or improving existing features to meet the evolving needs of the users.
3. **Performance Optimization:** The process of improving the speed, efficiency, and reliability of the software.
4. **Porting and Migration:** The process of adapting the software to run on new hardware or software platforms.
5. **Re-Engineering:** The process of improving the design and architecture of the software to make it more maintainable and scalable.
6. **Documentation:** The process of creating, updating, and maintaining the documentation for the software, including user manuals, technical specifications, and design documents.

## 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.

## Need for Maintenance

- Correct faults.
- Improve the design.
- Implement enhancements.
- Interface with other systems.
- Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
- Migrate legacy software.
- Retire software.
- Requirement of user changes.
- Run the code fast

### Challenges in Software Maintenance

- Lack of documentation
- Legacy code
- Complexity
- Changing requirements
- Interoperability issues
- Lack of test coverage
- Lack of personnel
- High-Cost

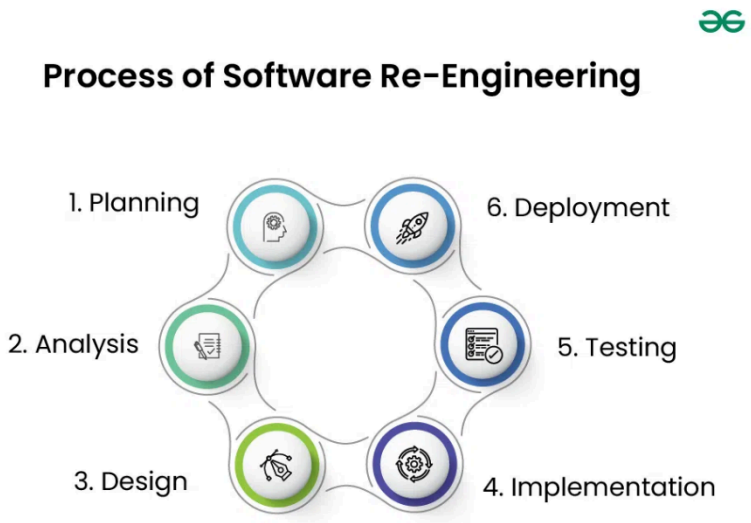
# # RE- ENGINEERING

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

1. 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.
2. 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.
3. It involves reusing the existing software artifacts, such as code, design, and documentation, and transforming them to meet new or updated requirements.

**Process of Software Re-engineering**

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




***Process of Software Re-engineering***

1. **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.

2. **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.
3. **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.
4. **Implementation:** The next step is to implement the changes by modifying the existing code, adding new features, and updating the documentation and other artifacts.
5. **Testing:** Once the changes have been implemented, the software system needs to be tested to ensure that it meets the new requirements and specifications.
6. **Deployment:** The final step is to deploy the re-engineered software system and make it available to end-users.

## Why Perform Re-engineering?

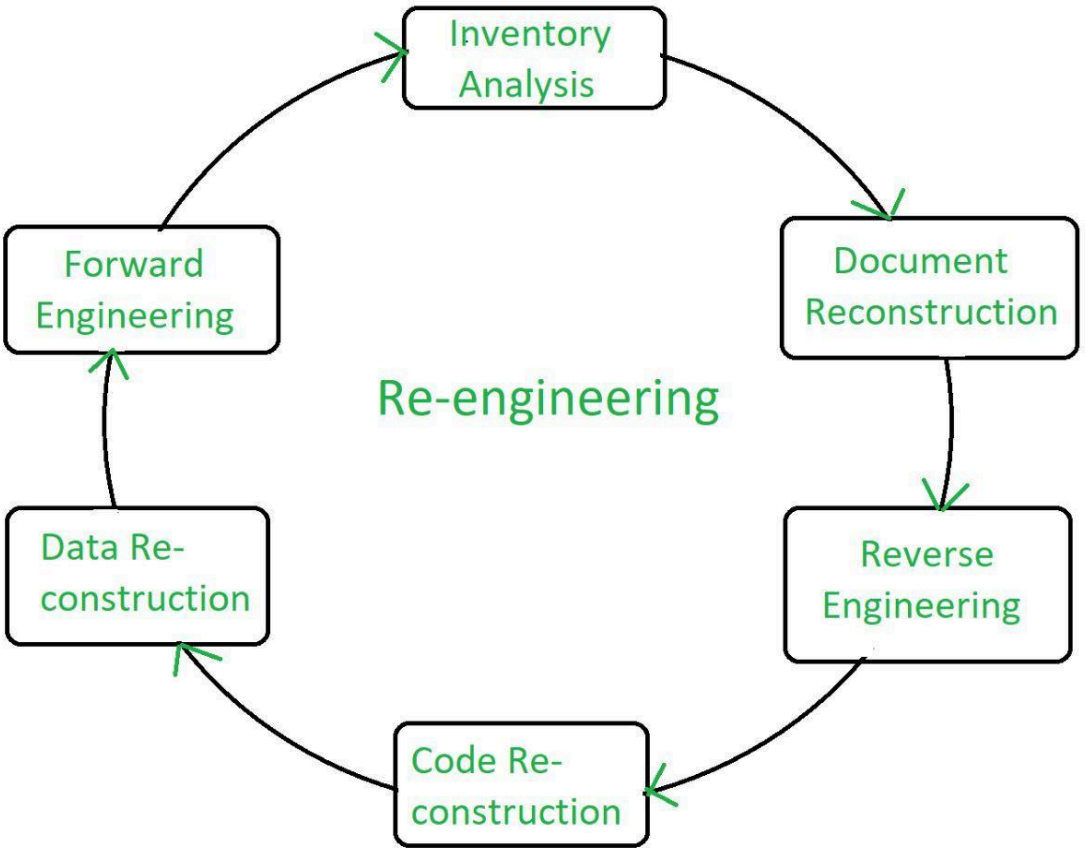
Re-engineering can be done for a variety of reasons, such as:

1. **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.
2. **To add new features:** Re-engineering can be used to add new features or functionality to existing software.
3. **To support new platforms**: Re-engineering can be used to update existing software to work with new hardware or software platforms.
4. **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.

5. **To meet new regulations and compliance**: Re-engineering can be done to ensure that the software is compliant with new regulations and standards.
6. **Improving software quality:** Re-engineering can help improve the quality of software by eliminating defects, improving performance, and enhancing reliability and maintainability.
7. **Updating technology:** Re-engineering can help modernize the software system by updating the technology used to develop, test, and deploy the system.
8. **Enhancing functionality:** Re-engineering can help enhance the functionality of the software system by adding new features or improving existing ones.
9. **Resolving issues:** Re-engineering can help resolve issues related to scalability, security, or compatibility with other systems.

## Steps involved in Re-engineering

1. **Inventory Analysis**
2. **Document Reconstruction**
3. **Reverse Engineering**
4. **Code Reconstruction**
5. **Data Reconstruction**
6. **Forward Engineering**


Inventory Analysis
Document Reconstruction
Reverse Engineering
Code Re-construction
Data Re-construction
Forward Engineering
Re-engineering