Software Engineering Project Management

Name of Subject Teacher : Prof.Sadaf Zama

SEMESTER : VI Sem

UNIT : I

Course Title : Software Engineering and Project Management (MDM II)
Semester : IV
Course Category : MDM2
Course Code : 23BTITL 204
Teaching Mode : Offline
TAE :10
CAE :15
TOTAL : 75 Marks
Duration of ESE : 3 Hrs.

Course Objectives

1. Understand the phases in a software project

2. Understand fundamental concepts of requirements engineering and Analysis Modeling.

3. Understand the major considerations for enterprise integration and deployment.

4.Learn various testing and maintenance measures

Course Outcomes

CO1: Identify the key activities in managing a software project

CO2: Compare different process models and design techniques.

CO3: Understand the concepts of requirements engineering and Analysis. Modeling.

CO4: Apply quality management and project management procedure for software design and deployment.

CO5: Remember various software testing and planning approaches.

UNIT 1:

Contents:

SOFTWARE PROCESS

1. Introduction to Software Engineering,
2. Software Process,
3. Perspective and Specialized Process Models,
4. Agile models,
5. Software Project Management,
6. Project Scheduling – Scheduling,
7. Earned Value Analysis.

Introduction to Software Engineering:

- Software Engineering is the process of planning, designing, building, testing, and maintaining software in a systematic and organized way.
- Instead of just writing code randomly, software engineering helps us build software that works well, is reliable, and is easy to maintain.

What is Software?

- Software is a set of programs and instructions that tell a computer what to do.
  Examples:

Mobile apps (WhatsApp, Instagram)
Websites
Banking systems
College management systems

What is Engineering?

- Engineering means using scientific methods and best practices to solve real-world problems efficiently.

what is Software Engineering?

Software Engineering = Writing software using proper rules, steps, and methods
It helps to:
Reduce errors
Save time and cost
Improve quality
Make software easy to update in the future

Why do we need Software Engineering?

If software is developed without planning:

•It may have many bugs

•It may not meet user needs

•It may be difficult to modify later

Software engineering helps to:

•Understand user requirements

•Plan before coding

•Deliver good-quality software on time

Main Activities in Software Engineering

Requirement Analysis – Understand what the user wants

Design – Plan how the software will work

Development (Coding) – Write the actual program

Testing – Check for errors and fix them

Maintenance – Update and improve the software

Example (Simple)

Think of building a house

•First, you decide what kind of house you want (requirements)

•Then make a blueprint (design)

•Then build the house (coding)

•Inspect it for problems (testing)

•Repair and upgrade when needed (maintenance)

Software engineering works in the same way.

QUESTION BANK:

1.Explain Software Engineering and Software Process in detail.

2.Describe Perspective and Specialized Process Models with neat diagrams.

3.Explain Agile Models in detail. Discuss their principles and benefits.

4.Explain Software Project Management and its key activities.

5.Describe Project Scheduling in software project management.

6.Explain Earned Value Analysis (EVA) in detail. Derive and explain:

- Planned Value (PV)
- Earned Value (EV)
- Actual Cost (AC)
- Schedule Variance (SV)
- Cost Variance (CV)

7.Compare Agile models and traditional software process models.

1.Explain Software Engineering and Software Process in detail.

Software Engineering

- Software Engineering is a systematic way of planning, designing, developing, testing, and maintaining software so that it works correctly, is reliable, and can be easily modified.
- Software Engineering means building software in the right way, not in a hurry or randomly.

Why Software Engineering is Needed

Without software engineering:

- Software may have many errors
- Cost and time may increase
- Software may not satisfy user needs
- Maintenance becomes difficult

Software engineering helps to:

- Reduce errors
- Improve quality
- Save time and money
- Deliver reliable software

Objectives of Software Engineering

- To develop correct and reliable software
- To deliver software on time and within budget
- To make software easy to maintain and upgrade
- To meet user requirements

Key Activities of Software Engineering

1.Requirement Analysis – Understanding what the user wants
2.Design – Planning how the software will work
3.Coding – Writing the program
4.Testing – Finding and fixing errors
5.Maintenance – Updating and improving software

Example

Just like building a house:
Decide requirements (number of rooms)
Make a blueprint
Construct the house
Inspect it
Repair and upgrade when needed
Software engineering follows the same idea.

Software Process

A software process is a set of steps or activities that are followed to develop software from start to finish.
Software process is a step-by-step method to create software.

Purpose of Software Process

- Provides a clear roadmap for development
- Helps in managing time and cost
- Ensures quality and consistency
- Makes work easier to control and track

Main Activities in a Software Process

Specification – Decide what the software should do
Design & Development – Plan and build the software
Validation – Check whether the software works correctly
Evolution – Modify and improve the software after delivery

Types of Software Process Models

Waterfall Model
Agile Model
Spiral Model
Incremental Model
Each model follows a different way of organizing these activities.

Relationship Between Software Engineering and Software Process

•Software Engineering is the overall discipline
•Software Process is the method used within software engineering
Software engineering uses software processes to build software effectively.

Difference:

Software Engineering	Software Process
Overall discipline	Step-by-step method
Focuses on quality and management	Focuses on development steps
Uses tools, techniques, models	Defines how work is done

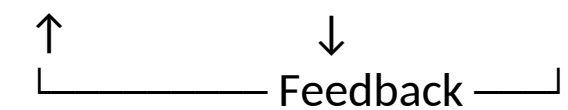
2.Describe Perspective and Specialized Process Models with neat diagrams.

1. Perspective Process Models

- A Perspective Process Model explains software development from a particular viewpoint (perspective).
- Each model focuses on one important aspect of development such as planning, risk, reuse, or user feedback.
- It does not show the entire process in detail, but highlights how development looks from one angle.
- “Same software process, different ways to look at it.”

Common Perspectives
Workflow perspective – order of activities
Data-flow perspective – how data moves
Role perspective – who does what

Example: Iterative Perspective
Software is developed in repeated cycles, and each cycle improves the product.
Neat Diagram



Advantages

•Easy to understand
•Helps focus on a specific concern
•Improves clarity for managers and developers

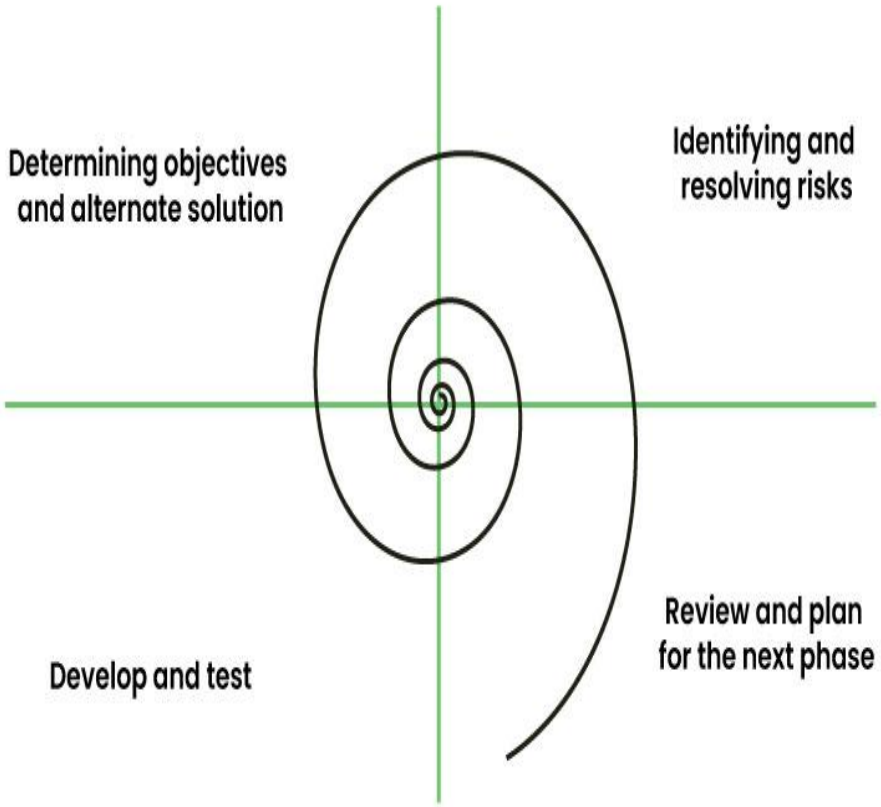
Disadvantages

•Does not show full technical details
•Needs combination with other models

2. Specialized Process Models

- Specialized Process Models are used for specific types of software projects where normal models are not suitable.
- These models are chosen based on project needs such as risk, reuse, or rapid delivery.
- High risk projects
- Projects needing fast delivery
- Projects based on reusable components

1.Spiral Model



Spiral model

Planning
↑
Risk Analysis → Engineering
↓
Evaluation

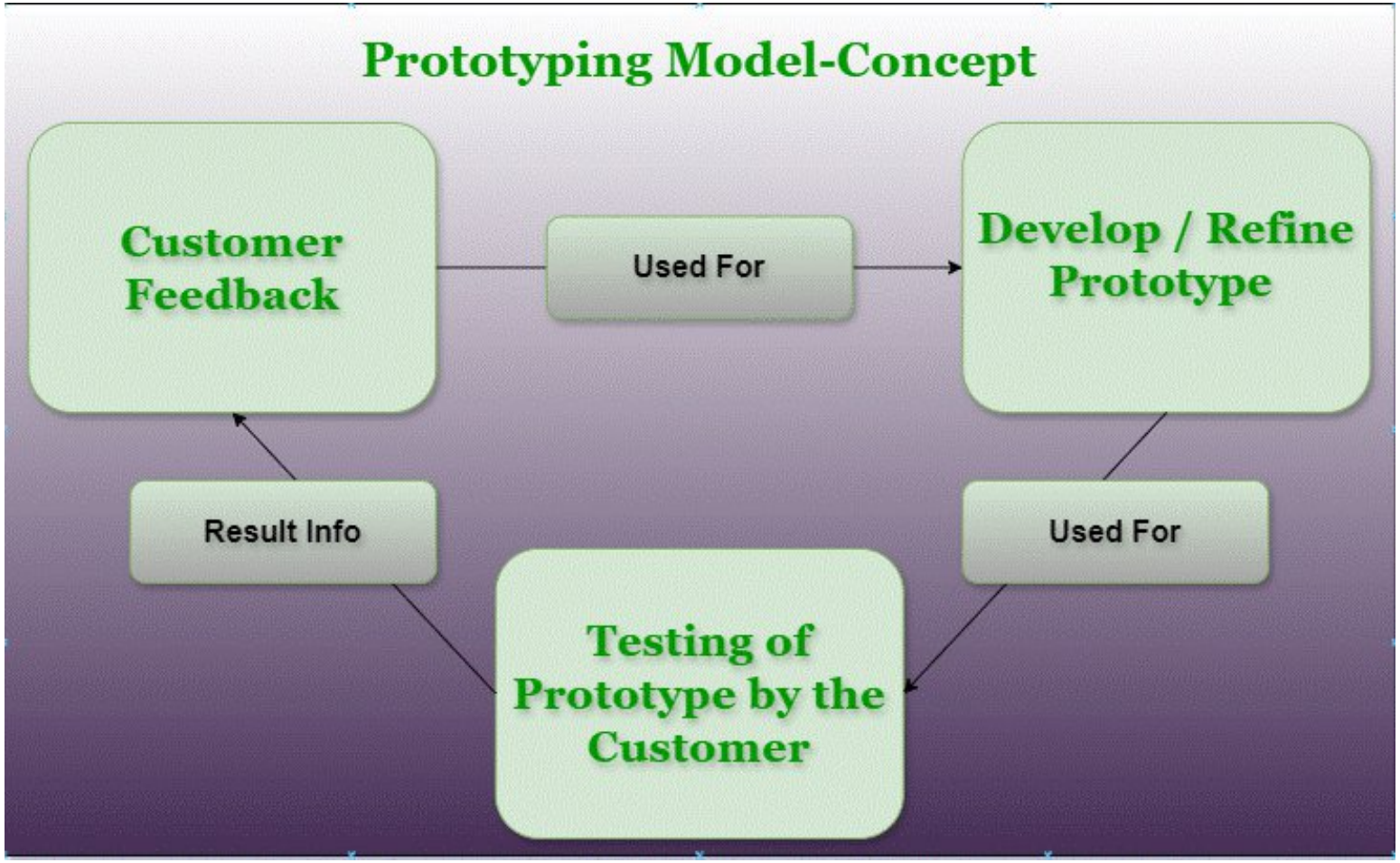
- It is a software development process model.
- This model has characteristics of both iterative and waterfall models.
- This model is used in projects which are large and complex.
- This model was named spiral because if we look at its figure, it looks like a spiral, in which a long curved line starts from the center point and makes many loops around it.
- The number of loops in the spiral is not decided in advance but it depends on the size of the project and the changing requirements of the user.
- We also call each loop of the spiral a phase of the software development process.
- In Spiral Model the entire process of software development is described in four phases which are repeated until the project is completed.
- A software project goes through these loops again and again in iterations.
- After each iteration a more and more complete version of the software is developed.
- The most special thing about this model is that risks are identified in each phase and they are resolved through prototyping.
- This feature is also called Risk Handling.

Advantages

•Good for large and risky projects

•Early problem detection

2.Prototyping Model



User Requirements

↓

Prototype

↓

User Feedback

↓

Final System

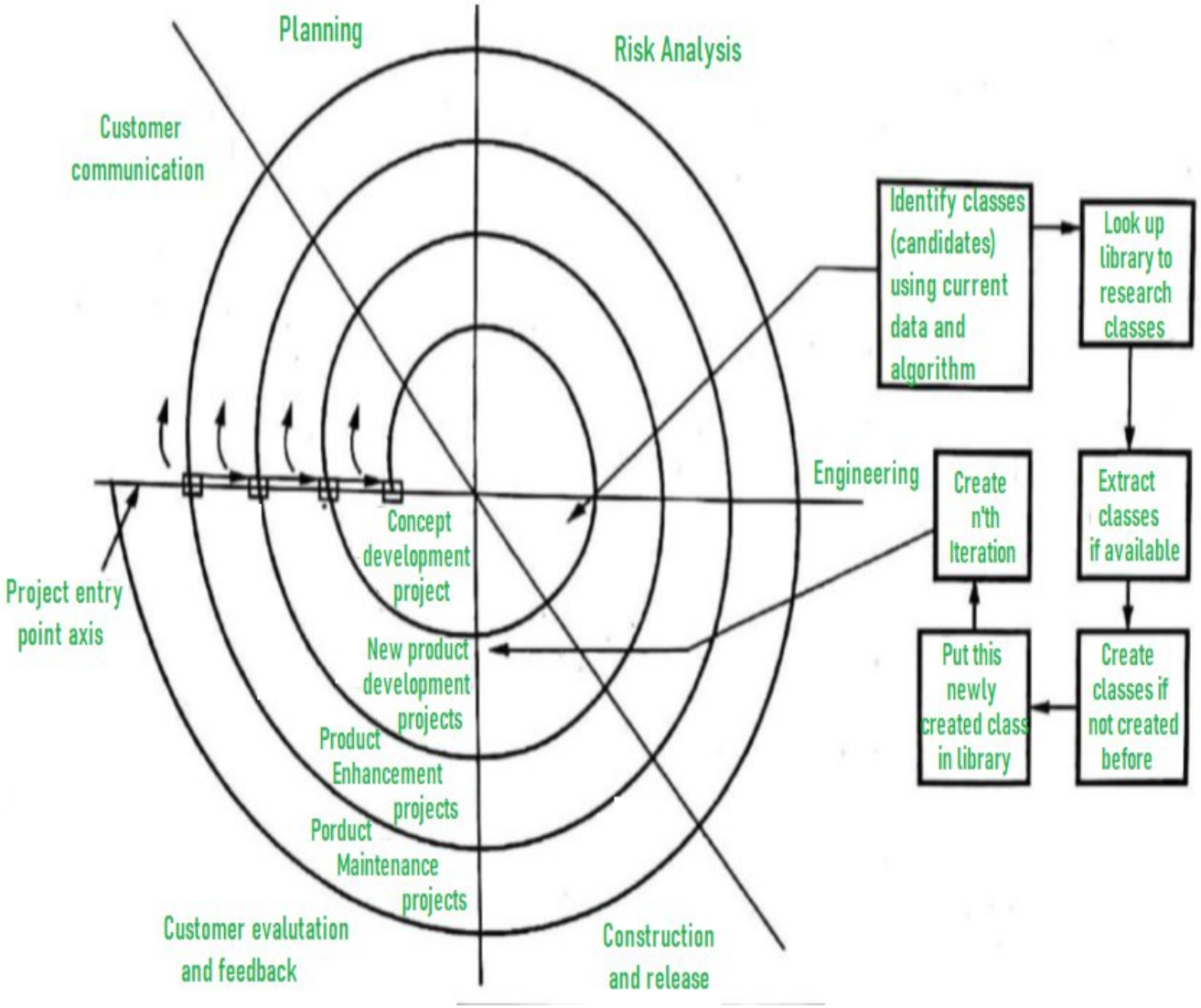
- In this model, a prototype of the end product is first developed, tested, and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.
- In this process model, the system is partially implemented before or during the analysis phase thereby allowing the customers to see the product early in the life cycle.
- The process starts by interviewing the customers and developing the incomplete high-level paper model.
- This document is used to build the initial prototype supporting only the basic functionality as desired by the customer.
- Once the customer figures out the problems, the prototype is further refined to eliminate them.
- The process continues until the user approves the prototype and finds the working model to be satisfactory.

Advantages

- Clear requirements
- High user satisfaction

3. Component-Based Development Model

- The component-based assembly model uses object-oriented technologies.
- In object-oriented technologies, the emphasis is on the creation of classes.
- Classes are the entities that encapsulate data and algorithms.
- In component-based architecture, classes (i.e., components required to build application) can be uses as reusable components.
- This model uses various characteristics of spiral model.
- This model is evolutionary by nature.
- Hence, software development can be done using iterative approach.
- multiple classes can be used.
- These classes are basically the prepackaged components.



Component Library

↓ ↓

Component A Component B

↓

System Integration

The model works in following manner:
Step-1: First identify all the required candidate components, i.e., classes with the help of application data and algorithms.
Step-2: If these candidate components are used in previous software projects then they must be present in the library.
Step-3: Such preexisting components can be excited from the library and used for further development.
Step-4: But if the required component is not present in the library then build or create the component as per requirement.
Step-5: Place this newly created component in the library. This makes one iteration of the system.
Step-6: Repeat steps 1 to 5 for creating n iterations, where n denotes the number of iterations required to develop the complete application.

Advantages
- Saves time and cost
- Reliable components

Characteristics
- Uses object-oriented technology.
- Components and classes encapsulate both data and algorithms.
- Components are developed to be reusable.
- Paradigm similar to spiral model, but engineering activity involves components.
- The system produced by assembling the correct components.

3.Explain Agile Models in detail. Discuss their principles and benefits.

- The Agile Model was primarily designed to help a project adapt quickly to change requests.
- So, the main aim of the Agile model is to facilitate quick project completion.
- To accomplish this task, it's important that agility is required.
- Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project.
- Also, anything that is a waste of time and effort is avoided.
- The Agile Model refers to a group of development processes.
- These processes share some basic characteristics but do have certain subtle differences among themselves.

Steps in the Agile Model

- The Agile Model is a combination of iterative and incremental process models.
- The phases involve in Agile Model are:

1.Requirement Gathering
2.Design the Requirements
3.Construction / Iteration
4.Testing / Quality Assurance
5.Deployment
6.Feedback

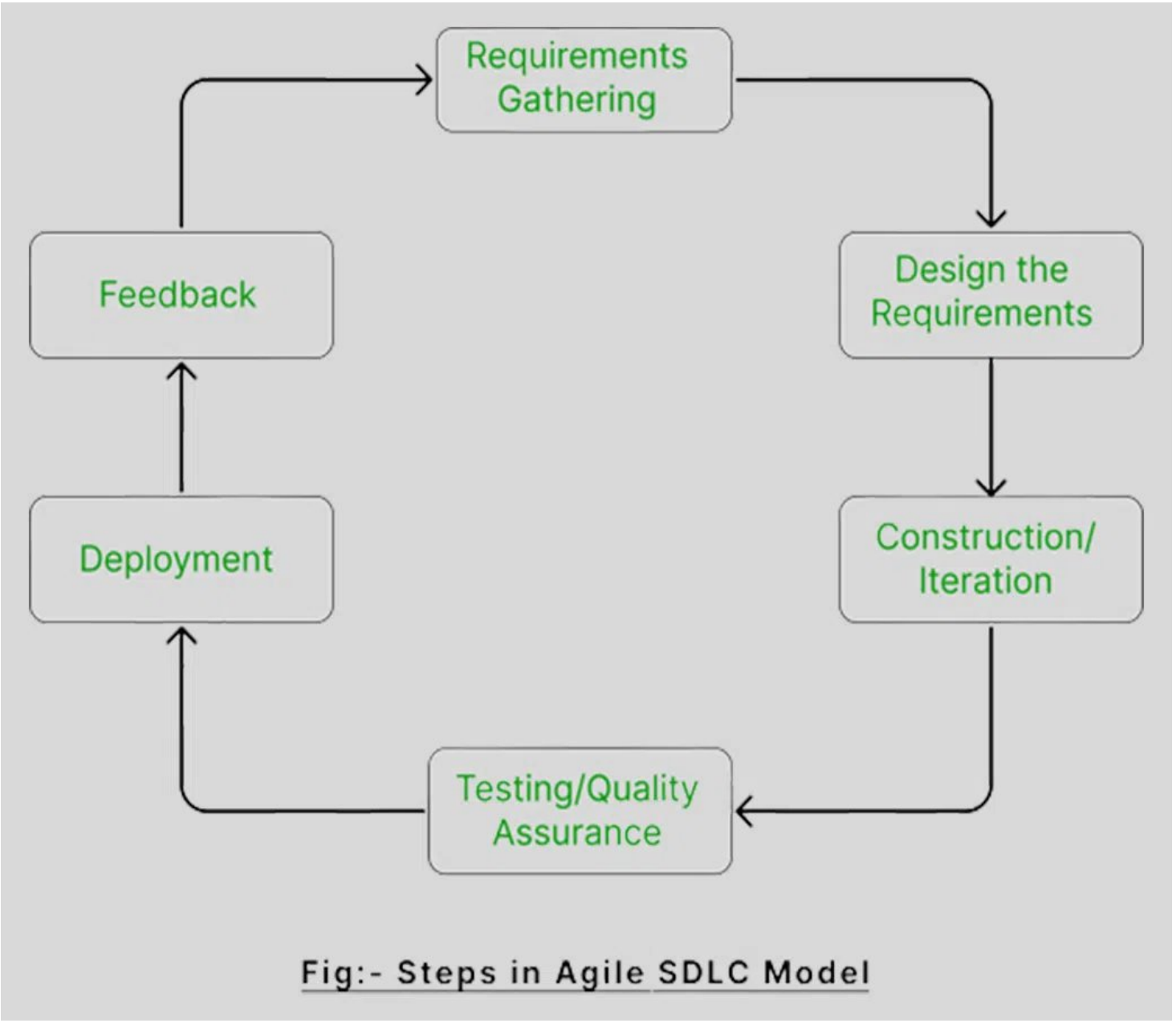


Fig:- Steps in Agile SDLC Model

Plan → Design → Develop → Test → Review

↑ ↓

└ Feedback ┘

1. Requirement Gathering

- In this step, the development team must gather the requirements, by interaction with the customer.
- development team should plan the time and effort needed to build the project.
- Based on this information you can evaluate technical and economical feasibility.

•Meet with the customer to really understand their needs and what they expect from the software.

•Identify the key requirements and business goals to make sure everyone is on the same page.

•Estimate how much time and effort it will take to develop the software.

•Assess if the project is technically possible and whether it's worth the investment from both a technical and economic standpoint.

2. Design the Requirements

- In this step, the development team will use user-flow-diagram or high-level UML Diagrams to show the working of the new features and show how they will apply to the existing software.
- Wireframing and designing user interfaces are done in this phase.

Designing the system:

- Once the requirements are gathered, the next step is to design the system’s overall architecture based on those needs.

- This helps to verify the software is structured in a way that meets the user's expectations.

Creating wireframes:
- Next, wireframes for the user interface (UI) are created.
- These are simple blueprints that show how the software will look and how users will interact with it, ensuring it's user-friendly and easy to navigate.

High-level design with UML diagrams:
- At this stage, high-level designs using UML (Unified Modeling Language) diagrams are created to visually represent the software's structure and how different parts will work together.

Prototyping for feedback:
- Prototypes are made to give stakeholders an early look at the software.
- This helps gather feedback early in the process and allows for adjustments before the full development begins.

3. Construction / Iteration
- In this step, development team members start working on their project, which aims to deploy a working product.
- Each cycle typically consist between **1-4 weeks**, and at the end, the team delivers a working version of the software.

Development of Features:
- The team works on the features identified during the requirement and design phases.

Coding and Implementation:

- New functionalities are coded and integrated into the software based on the goals for that specific iteration.

Delivering a Working Product:

- After each iteration, a usable version of the software is ready.

Incremental Improvement:

- With every cycle, the software is enhanced, adding more features and refining existing ones.

4. Testing / Quality Assurance

Testing involves Unit Testing, Integration Testing, and System Testing, Which help in the agile development models:

Unit Testing:

- Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own.
- Unit testing is used to test individual blocks (units) of code.

Integration Testing:

- Integration testing is used to identify and resolve any issues that may arise when different units of the software are combined.

System Testing:

- Goal is to ensure that the software meets the requirements of the users and that it works correctly in all possible scenarios.

5. Deployment

- In this step, the development team will deploy the working project to end users.
- Once an iteration is finished and fully tested, the software is ready to be released to the end users.
- In Agile, deployment isn't a one-time event—it's an ongoing process.
- Updates and improvements are rolled out regularly, making sure the software keeps evolving and getting better with each release.

•Deploy the software to a test or live environment so that it can be used by customers or end-users.

•Make the software accessible to users, verifying they can start using it as expected.

•Verify the deployment goes smoothly and fix any issues that come up quickly

6. Feedback

- This is the last step of the **Agile Model.**
- In this, the team receives feedback about the product and works on correcting bugs based on feedback provided by the customer.
- Take feedback from customers, users, and stakeholders after each iteration.
- Understand how well the product meets user needs and identify areas for improvement.
- Check for bugs or issues that need fixing.
- Make adjustments to the development plan based on feedback to improve the product further.

Popular Agile Models (Types)

1. Scrum

- Work is done in short cycles (sprints) of 2–4 weeks
- Team meets daily for a short discussion
- Product improves after every sprint

Backlog → Sprint → Working Software → Feedback

2. Extreme Programming (XP)

- Focus on high-quality code
- Frequent testing and continuous improvement
- Very strong customer involvement

3. Kanban

- Work is shown on a Kanban board
- Limits work in progress
- Focus on continuous delivery

To Do → In Progress → Testing → Done

Principles of the Agile Model

1.Our highest priority is to satisfy the client through early and continuous delivery of valuable computer software.

2.Welcome dynamic necessities, even late in development. Agile Processes harness modification for the customer's competitive advantage.

3.Deliver operating computer software often, from a pair of weeks to a couple of months, with a preference to the shorter timescale.

4.Business individuals and developers should work along daily throughout the project.

5.The build comes around actuated people. offer them the setting and support they have, and trust them to urge the task done.

6.the foremost economical and effective methodology of conveyancing info to and among a development team is face-to-face speech.

7.Working with computer software is the primary life of progress.

8.Agile processes promote property development. The sponsors, developers, and users will be able to maintain a relentless pace indefinitely.

9.Continuous attention to technical excellence and smart style enhances nimbleness.

10.Simplicity—the art of maximizing the number of work not done—is essential.

11.the most effective architectures, necessities, and styles emerge from self–organizing groups.

12.At regular intervals, the team reflects on a way to become simpler, then tunes and adjusts its behavior consequently.

Advantages of the Agile Model

- Working through Pair programming produces well-written compact programs which have fewer errors as compared to programmers working alone.
- It reduces the total development time of the whole project.
- Agile development emphasizes face-to-face communication among team members, leading to better collaboration and understanding of project goals.
- Customer representatives get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.
- Agile development puts the customer at the center of the development process, ensuring that the end product meets their needs.

Disadvantages of the Agile Model

- The lack of formal documents creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
- It is not suitable for handling complex dependencies.
- The agile model depends highly on customer interactions so if the customer is not clear, then the development team can be driven in the wrong direction.

- Agile development models often involve working in short sprints, which can make it difficult to plan and forecast project timelines and deliverables.
- This can lead to delays in the project and can make it difficult to accurately estimate the costs and resources needed for the project.
- Agile development models require a high degree of expertise from team members, as they need to be able to adapt to changing requirements and work in an iterative environment.
- This can be challenging for teams that are not experienced in agile development practices and can lead to delays and difficulties in the project.
- Due to the absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.

Benefits of Agile Models

1. Faster Delivery

Software is delivered **quickly** in small parts

2. Better Quality

Continuous testing reduces errors

3. Customer Satisfaction

Customer feedback is included regularly

4. Flexibility

Easy to **change requirements**

5. Reduced Risk

Problems are found **early**

6. Better Team Communication

Daily interaction improves understanding

4.Explain Software Project Management and its key activities.

- Software Project Management (SPM) is a proper way of planning and leading software projects.
- It is a part of project management in which software projects are planned, implemented, monitored, and controlled.
- Software Project Management (SPM) is all about planning, organizing, and observing the software development process to make sure the project is completed successfully, on time, and within budget.
- It involves tasks like planning, defining the project scope, calculating how much time and resources are needed, scheduling tasks, allocating resources, and tracking progress.
- The ultimate goal is to deliver a high-quality software product that meets the needs and expectations of the users.

Need for Software Project Management

- Software is a non-physical product.
- Software Development is a new stream in business, and there is very little experience in building software products.
- Most of the software products are made to fit clients' requirements.
- The most important thing is that basic technology changes and advances so frequently and rapidly that the experience of one product may not be applied to another one.
- Such types of business and environmental constraints increase Risk in Software Development hence it is essential to manage software projects efficiently.
- It is necessary for an organization to deliver quality products, keep the cost within the client's budget constraint, and deliver the project as per schedule.
- Hence, in order, software project management is necessary to incorporate user requirements along with budget and time constraints.

Types of Management in SPM

1. Conflict Management

- Conflict management is the process to restrict the negative features of conflict while increasing the positive features of conflict.
- The goal of conflict management is to improve learning and group results including efficacy or performance in an organizational setting.
- Properly managed conflict can enhance group results.

2. Risk Management

- Risk management is the analysis and identification of risks that is followed by synchronized and economical implementation of resources to minimize, operate and control the possibility or effect of unfortunate events or to maximize the realization of opportunities.

3. Requirement Management

- It is the process of analyzing, prioritizing, tracking, and documenting requirements and then supervising change and communicating to pertinent stakeholders. It is a continuous process during a project.

4. Change Management

- Change Management is a systematic approach to dealing with the transition or transformation of an organization's goals, processes, or technologies.
- The purpose of change management is to execute strategies for effecting change, controlling change, and helping people to adapt to change.

5. Software Configuration Management

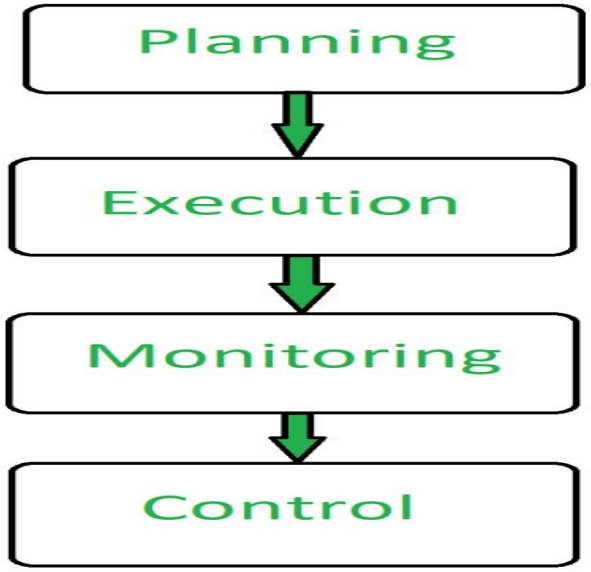
- Software configuration management is the process of controlling and tracking changes in the software, part of the larger cross-disciplinary field of configuration management.
- Software Configuration Management includes revision control and the inauguration of baselines.

6. Release Management

- Release Management is the task of planning, controlling, and scheduling the built-in deploying releases.
- Release management ensures that the organization delivers new and enhanced services required by the customer while protecting the integrity of existing services.

Aspects of Software Project Management

- Software project management involves several key aspects that help guide the development process from start to finish.
- These aspects verify that the project stays on track, meets user needs, and is delivered within time and budget constraints.
- Here are the main aspects of software project management:



1. Planning

- The Software Project Manager lays out the complete project's blueprint.
- The project plan will outline the scope, resources, timelines, techniques, strategy, communication, testing, and maintenance steps.
- SPM can aid greatly here.

2. Leading

- A software project manager brings together and leads a team of engineers, strategists, programmers, designers, and data scientists.
- Leading a team necessitates exceptional communication, interpersonal, and leadership abilities.
- One can only hope to do this effectively if one sticks with the core SPM principles.

3. Execution

- SPM comes to the rescue here also as the person in charge of software projects (if well versed with SPM/Agile methodologies) will ensure that each stage of the project is completed successfully.
- measuring progress, monitoring to check how teams function, and generating status reports are all part of this process.

4. Time Management

- Abiding by a timeline is crucial to completing deliverables successfully.
- This is especially difficult when managing software projects because changes to the original project charter are unavoidable over time.
- To assure progress in the face of blockages or changes, software project managers ought to be specialists in managing risk and emergency preparedness.
- This Risk Mitigation and Management is one of the core tenets of the philosophy of SPM.

5. Budget

- Software Project Managers, like conventional project managers, are responsible for generating a project budget and adhering to it as closely as feasible, regulating spending, and reassigning funds as needed.
- SPM teaches us how to effectively manage the monetary aspect of projects to avoid running into a financial crunch later on in the project.

6. Maintenance

- Software project management emphasizes continuous product testing to find and repair defects early, tailor the end product to the needs of the client, and keep the project on track.
- The Software Project Manager makes ensuring that the product is thoroughly tested, analyzed, and adjusted as needed. Another point in favor of SPM.

Drawbacks of Software Project Management

1. Costs are High

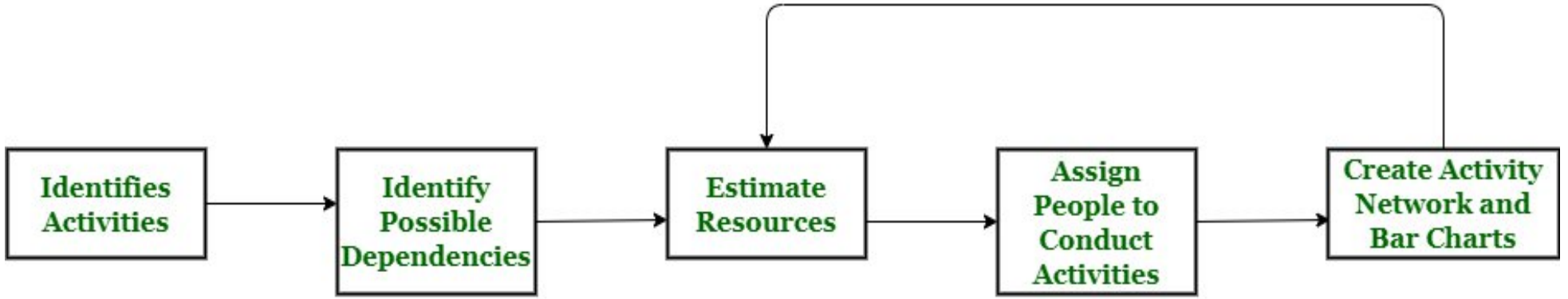
2. Complexity will be increased

3. Overhead in Communication

4. Lack of Originality

5.Describe Project Scheduling in software project management.

- A schedule in your project's time table actually consists of sequenced activities and milestones that are needed to be delivered under a given period of time.
- Project schedule simply means a mechanism that is used to communicate and know about that tasks are needed and has to be done or performed and which organizational resources will be given or allocated to these tasks and in what time duration or time frame work is needed to be performed.
- Effective project scheduling leads to success of project, reduced cost, and increased customer satisfaction.
- Scheduling in project management means to list out activities, deliverables, and milestones within a project that are delivered.
- It contains more notes than your average weekly planner notes.
- The most common and important form of project schedule is Gantt chart



Project Scheduling Process

•**Process** :

•The manager needs to estimate time and resources of project while scheduling project.

• All activities in project must be arranged in a coherent sequence that means activities should be arranged in a logical and well-organized manner for easy to understand.

•Initial estimates of project can be made optimistically which means estimates can be made when all favorable things will happen and no threats or problems take place.

• The total work is separated or divided into various small activities or tasks during project schedule.

•Then, Project manager will decide time required for each activity or task to get completed.

•Even some activities are conducted and performed in parallel for efficient performance.

• The project manager should be aware of fact that each stage of project is not problem-free.

• Problems arise during Project Development Stage :People may leave or remain absent during particular stage of development.

•Hardware may get failed while performing.

•Software resource that is required may not be available at present, etc.

•The project schedule is represented as set of chart in which work-breakdown structure and dependencies within various activities are represented.

•To accomplish and complete project within a given schedule, required resources must be available when they are needed.

•Therefore, resource estimation should be done before starting development.

•Resources required for Development of Project :

•Human effort

•Sufficient disk space on server

•Specialized hardware

•Software technology

•Travel allowance required by project staff, etc.

•**Advantages of Project Scheduling :**

•There are several advantages provided by project schedule in our project management:

•It simply ensures that everyone remains on same page as far as tasks get completed, dependencies, and deadlines.

•It helps in identifying issues early and concerns such as lack or unavailability of resources.

•It also helps to identify relationships and to monitor process.

•It provides effective budget management and risk mitigation.

6.Explain Earned Value Analysis (EVA) in detail. Derive and explain:

- Planned Value (PV)
- Earned Value (EV)
- Actual Cost (AC)
- Schedule Variance (SV)
- Cost Variance (CV)

- DefenseEarned Value Analysis (EVA) is also called "Budget cost of work performed".
- It is considered a refinement of the cost-monitoring technique.
- This analysis was first carried out USA's Department of Defense (DOD).
- In this analysis, a "value" is assigned to each track or work package based on the expenditure forecast.
- The value assigned is known as the "planned value (PV)".
- The work that has not yet begun is given a value known as the "earned value of zero".
- The total value credited to a project is called "earned value(EV)", which is also represented as "money value".

Methods For Earned Value Analysis

0/100 Technique:

- The technique where a task is assigned a value of zero until such time that is completed when it is given a value of 100% of the budgeted value.

50/50 Technique:

- The technique in which a task is assigned a 50% value as soon as it is started and then given a value.

75/25 Technique:

- The technique where a task is assigned 75% on starting and 25% on completion.

Milestone Technique:

- The technique where a task is given a value based on the achievement of milestones that have been assigned values as part of the original budget plan.

Percentage Complete:

- In some cases, there may be a way of objectively measuring the amount of work completed.

Stages in Earned Value Analysis

Creating the baseline budget:

- This is the first stage in setting up EVA.
- This budget is based on the project plan.
- It predicts the earned value through time.
- Normally, it is measured in person hours or workdays,
- example: in a software development project.

Monitoring Earned Value:

- The second stage is monitoring the earned value as the project progresses.
- This is achieved by monitoring the completion of each task.
- Actual cost(AC) is the actual cost of each task and it can be analyzed and collected.

Schedule Variance(SV):

This is the third stage which is measured in cost as EV-PV which is the deviation between planned work and completed work.

Example: Consider these values,

PV =40000

EV=35000

SV=35000-40000 = -5000

Here the calculated SV value is negative and hence we conclude that the project is behind the original schedule.

•**Time variance(TV):**

•The difference between the current time and the time when the achievement of the earned value was planned to occur.

•**Cost Variance(CV):**

• This value is the difference between the actual cost and the earned value.

•Using this value, we can estimate the accuracy of the original cost scheduled for the project.

•If the CV values are found to be negative, we conclude the project is over cost.

Advantages of Earned Value Analysis (EVA)

Project Performance Measurement:

- EVA provides a comprehensive method for measuring and assessing the performance of a project.
- It helps project managers gain a clear understanding of how well a project is progressing in terms of cost and schedule.

Objective Performance Metrics:

- EVA relies on objective metrics, making it less susceptible to subjective interpretations.
- This can lead to more accurate assessments of project performance.

Integration of Cost and Schedule:

- EVA combines cost and schedule performance, allowing project managers to see the relationship between these two critical aspects of project management.
- This integration can help in identifying issues early and making informed decisions.

Early Issue Identification:

- EVA can highlight problems in project execution early, enabling project managers to take corrective actions promptly.
- This can prevent cost overruns and schedule delays.

Benchmarking:

- EVA allows for benchmarking project performance against planned targets and historical data.
- It helps project managers assess whether their project is on track compared to similar projects.

Effective Communication:

- EVA provides a standardized way to communicate project performance to stakeholders.
- Charts and reports generated from EVA data can make it easier for stakeholders to understand the project's status.

Disadvantages of Earned Value Analysis (EVA)

Complexity:

- EVA involves complex calculations and terminology, which can be challenging for project teams to understand, especially for smaller projects or teams with limited expertise.

Resource-Intensive:

- Implementing EVA requires tracking detailed data and maintaining comprehensive records.
- This can be resource-intensive, and some organizations may lack the necessary tools or resources for effective EVA implementation.

Subjectivity in Earned Value Calculation:

- The "earned value" itself can be subject to interpretation, especially in situations where there are gray areas regarding the completion of work packages or milestones.

Assumption of Linear Progression:

- EVA assumes linear progression, which means that it may not work well for projects with irregular or non-linear progress patterns, such as research and development projects.

Time-Consuming:

- Calculating and updating EVA metrics can be time-consuming, which may not be suitable for projects that require quick decision-making and frequent changes.

Focus on Metrics vs. Problem Solving:

- Overemphasis on EVA metrics can sometimes lead to a focus on numbers rather than addressing the underlying issues causing performance problems.
- EVA combines scope, time, and cost into a single performance measurement system.
- It uses three basic values:

Planned Value (PV)
Earned Value (EV)
Actual Cost (AC)

- From these, two important variances are calculated:

Schedule Variance (SV)
Cost Variance (CV)

1. Planned Value (PV)

- Planned Value (PV) is the budgeted cost of work that was planned to be completed up to a specific time.
- It shows how much work should have been done according to the project plan.

Formula (Derivation)
\text{PV} = \text{Planned % of work} \times \text{Total Project Budget}

- PV is decided before the project starts
- It is based on the project schedule
- It does not depend on actual work done

Example
Total project budget = ₹1,00,000
Planned completion by week 5 = 40%

$$\text{PV} = 0.40 \times 1{,}00{,}000 = ₹40{,}000$$

Meaning: By week 5, work worth ₹40,000 should have been completed.

2. Earned Value (EV)

- Earned Value (EV) is the budgeted cost of the work that has actually been completed.
- It shows how much work is really done, measured in terms of budget.

Formula (Derivation)
\text{EV} = \text{Actual % of work completed} \times \text{Total Project Budget}

- EV depends on actual progress
- It converts completed work into monetary value

Example
Total project budget = ₹1,00,000
Actual work completed = 30%

$$\text{EV} = 0.30 \times 1{,}00{,}000 = ₹30{,}000$$

Meaning: Only work worth ₹30,000 has been completed.

3. Actual Cost (AC)

Actual Cost (AC) is the actual amount of money spent to complete the work done so far.

Formula

$$AC = \textit{Total actual expenditure till date}$$

Explanation

•AC comes from real expenses

•Includes labor cost, material cost, equipment cost, etc.

Example

•Money spent till week 5 = ₹35,000

$$AC = ₹35{,}000$$

Meaning: ₹35,000 has actually been spent.

4. Schedule Variance (SV)

Schedule Variance (SV) shows whether the project is ahead of schedule or behind schedule.

Formula (Derivation)

$$SV = EV - PV$$

Explanation

SV compares work completed with work planned

It is measured in cost units

Interpretation

SV = 0 → On schedule

SV > 0 → Ahead of schedule

SV < 0 → Behind schedule

Example

$$SV = 30{,}000 - 40{,}000 = -10{,}000$$

Meaning: Project is behind schedule by ₹10,000 worth of work.

5. Cost Variance (CV)

Cost Variance (CV) shows whether the project is under budget or over budget.

Formula (Derivation)

$$CV = EV - AC$$

Explanation

CV compares value of work done with actual cost

Indicates cost efficiency

Interpretation

CV = 0 → On budget

CV > 0 → Under budget

CV < 0 → Over budget

Example

$$CV = 30{,}000 - 35{,}000 = -5{,}000$$

Meaning: Project is over budget by ₹5,000.

7.Compare Agile models and traditional software process models.

Traditional Software Development	Agile Software Development
It is used to develop simple software.	It is used to develop complicated software.
In this methodology, testing is done once the development phase is completed.	In this methodology, testing and development processes are performed concurrently.
It follows a linear organizational expectation structure.	It follows an iterative organizational structure.
It provides less security.	It provides high security.
Client involvement is less as compared to Agile development.	Client involvement is high as compared to traditional software development.

It provides less functionality in the software.	It provides all the functionality needed by the users.
It supports a fixed development model.	It supports a changeable development model.
It is used by freshers.	It is used by professionals.
Development cost is less using this methodology.	Development cost is high using this methodology.
It majorly consists of five phases.	It consists of only three phases.
It is less used by software development firms.	It is normally used by software development firms.
The expectation is favored in the traditional model.	Adaptability is favored in the agile methodology.

Traditional software development approaches are formal in terms of communication with customers.	Agile software development methodologies are casual. In other words, customers who work with companies that utilize Agile software development approaches are more likely to interact with them than customers who work with companies that use traditional software development methodology.
For starters, typical software development approaches employ a predictive approach. There is full specification and prediction of the software development processes because the product is produced through rigorous and explicit planning. Changes are not permitted in this technique because the time and cost of project development are fixed.	Here, a flexible approach is used as the software development approaches are founded on the notion of continual design improvement and testing relies on team and client feedback.
Examples •Office productivity suites •Data management software •Media players •Security programs	**Examples** •Sky •Phillips •JP Morgan Chase

Models based on Traditional Software Development-	Models based on Agile Software Development-
•Spiral Model •Waterfall Model •V Model •Incremental Model	•Scrum •Extreme Programming (XP) •Crystal •Dynamic Systems Development Method (DSDM) •Feature Driven Development (FDD) •Adaptive Software Development (ASD)