Unit-4

Software Evaluation and Costing

Software Evaluation

In software project management, **project evaluation** goes beyond simply celebrating launch day. It's a systematic and objective assessment of a project's performance throughout its lifecycle, enabling you to understand its effectiveness, identify areas for improvement, and inform future decisions.

What does it involve?

Project evaluation typically analyzes various aspects:

- **Scope and objectives:** Were all planned features delivered, and did they meet project goals?
- **Cost and budget:** Did the project stay within budget, and were resources efficiently utilized?
- **Schedule and timeline:** Were the project delivered on time, and were deadlines met?
- **Quality and performance:** Do the software meet quality standards and perform as expected?
- **Stakeholder satisfaction:** Were stakeholders satisfied with the project outcome and communication throughout?
- **Risk management:** Were potential risks identified and mitigated effectively?

Why is it important?

- **Identify successes and failures:** Learn from what worked well and what didn't to improve future projects.
- **Improve processes and methodologies:** Use insights to refine your project management approach.
- **Demonstrate value and accountability:** Show stakeholders the project's impact and justify future investments.
- **Motivate and reward team members:** Recognize achievements and address areas for improvement.

Benefits:

- **Enhanced decision-making:** Helps make informed choices based on data and evaluation.
- **Reduced risk of future failures:** By learning from past mistakes, you can proactively address potential issues.

- **Improved project management practices:** Evaluation findings can guide process improvements and team development.
- **Increased stakeholder confidence:** Transparency and data-driven evaluation foster trust and buy-in.

Challenges:

- **Defining meaningful metrics:** Choosing the right metrics to measure success depends on project goals and stakeholders.
- **Time and resource commitment:** Conducting a thorough evaluation requires dedicating resources and time.
- **Potential for bias:** Ensure objectivity and avoid subjective interpretations of data.

Ask the following in every topic while you create a software project or in any project management topic

Always perform 5W1H question and its answers

- What
- Where
- here
- When
- Why
- Which
- How

Who is Responsible for Project Evaluation?

1. Senior management
2. Project manager/coordinator
3. Team leader

When is project evaluation done?

Usually at the beginning of the project e.g., Step 0 of Step Wise Framework

What Assessment are performed for project evaluation / accepting or rejecting a project?

1. Strategic assessment
2. Technical assessment
3. Economic assessment

Notes By Anjali Kandel
NCMT college (Nepalgunj)

Strategic Assessment

Strategic Assessment is the critical first step of evaluating a potential project before diving into development. It's like checking your map and compass before embarking on a journey - it ensures you're headed in the right direction and that the journey is worth taking.

1. Used to assess whether a project fits in the long-term goal of the organization
2. Usually carried out by senior management
3. Needs a strategic plan that clearly defines the objectives of the organization
4. It evaluates individual projects against the strategic plan or the overall business objectives.

- For example

  What projects do Facebook prefer vs. LinkedIn and why???
- suitable for projects developed for use in the organization Portfolio management
- suitable for project developed for other companies by software

Technical Assessment

In software project management, **Technical Assessment** focuses on evaluating the feasibility and effectiveness of the chosen technology throughout the project lifecycle. It's like putting your tools and materials to the test before building your project.

Here's what a technical assessment typically involves:

1. Technology Suitability:

- Does the chosen technology meet the project's technical requirements and functionalities?
- Are there alternative technologies that might be more suitable or efficient?
- Can the technology integrate with existing systems and infrastructure?

2. Performance and Scalability:

- Can the technology handle the expected load and data volume?
- Is it scalable enough to accommodate future growth?
- What are the performance bottlenecks or limitations of the technology?

3. Security and Compliance:

- Does the technology meet security standards and regulations?

- Are there vulnerabilities or data security risks associated with the technology?
- How will the project address data privacy and compliance requirements?

4. Development and Maintenance Costs:

- What are the costs associated with development tools, licenses, and maintenance?
- Are there any hidden costs or ongoing subscriptions to consider?
- How will these costs impact the project budget?

5. Team Skills and Expertise:

- Does the team have the necessary skills and knowledge to work with the chosen technology?
- Are there any training needs or skill gaps that need to be addressed?
- How will the team stay up-to-date with the evolving technology landscape?

Technical assessments are conducted at various stages:

- **Project initiation:** During feasibility studies, to select the most suitable technology.
- **Design and development:** Throughout the development process, to identify and address technical challenges.
- **Deployment and testing:** To ensure the software performs as expected and meets technical requirements.

Who conducts the assessment?

Similar to strategic assessments, a team might be involved, including software architects, developers, technical leads, and external consultants specializing in the chosen technology.

Cost-Benefit Analysis

Cost-Benefit Analysis (CBA) is a structured approach to compare the projected **costs** of a project with its anticipated **benefits**, usually expressed in monetary terms. It's like balancing a scale to weigh the financial implications of a project before investing resources.

Here's what a CBA typically involves:

Identifying Costs:

- **Development costs:** Salaries, tools, licenses, infrastructure, etc.

- **Maintenance costs:** Ongoing upkeep, bug fixes, upgrades, etc.
- **Operational costs:** Data storage, training, support, etc.
- **Opportunity costs:** Resources dedicated to this project instead of others.

Identifying Benefits:

- **Increased revenue:** Generated through sales, subscriptions, etc.
- **Cost savings:** Improved efficiency, reduced errors, etc.
- **Intangible benefits:** Improved customer satisfaction, brand reputation, etc.

Quantifying Costs and Benefits:

- Assigning monetary values to costs and benefits whenever possible.
- Estimating future cash flows based on market analysis and historical data.
- Considering risks and uncertainties that might impact costs or benefits.

Calculating Metrics:

- **Net Present Value (NPV):** Present value of future benefits minus present value of costs. A positive NPV indicates a projected profit.
- **Internal Rate of Return (IRR):** Discount rate at which NPV equals zero. Represents the minimum acceptable return on investment.
- **Payback Period:** Time it takes for benefits to recoup initial costs. Shorter payback indicates faster return on investment.

Interpreting the Results:

- CBA results provide insights into the project's financial viability and potential return on investment (ROI).
- It helps decision-makers assess if the expected benefits outweigh the costs, justifying the project's go-ahead.
- Remember, CBA is not a guarantee of success, but a powerful tool for informed decision-making.

Benefits of CBA in Software Project Management:

- Improved resource allocation by prioritizing projects with higher ROI(Return on Investment).
- Increased stakeholder buy-in through transparent cost-benefit analysis.
- Early identification of financially risky projects, allowing for adjustments.
- Improved project scope and feature selection based on cost-benefit trade-offs.

Cash Flow Forecasting

Cash Flow Forecasting is like predicting the weather for your project's finances. It's the process of estimating the **incoming and outgoing cash flow** over a specific period, helping you understand when you'll have enough money to cover expenses and plan for potential financial shortfalls.

Here's what it involves:

1. Identifying Cash Inflows:

- Client payments for milestones or completed work
- Advance payments or funding received
- Revenue generated from the finished product (if applicable)

2. Identifying Cash Outflows:

- Salaries and benefits for development team
- Costs of tools, licenses, and infrastructure
- Vendor payments for services or materials
- Marketing and promotional expenses

3. Using historical data and assumptions:

- Past project expenses and income serve as a base reference.
- Assumptions about future client payments, development progress, and market trends are used to adjust the forecast.

4. Creating a forecast report:

- Typically shown in a spreadsheet or financial modeling software.
- Breaks down cash flow by week, month, or quarter.
- Highlights potential surpluses or deficits.

Benefits of Cash Flow Forecasting:

- **Improved financial planning:** Enables you to allocate resources effectively and avoid unplanned expenses.
- **Early identification of issues:** Helps you anticipate potential funding shortfalls and take corrective action.
- **Increased transparency:** Provides stakeholders with a clear picture of the project's financial health.

Notes By Anjali Kandel
NCMT college (Nepalgunj)

- **Enhanced decision-making:** Allows you to make informed choices about project scope, pricing, and resource allocation.

Cost-Benefit Evaluation Techniques

Cost-Benefit Evaluation (CBE) techniques are like different lenses you can use to analyze the financial viability of a software project. Each technique offers a unique perspective, helping you make informed decisions about investing your resources. Here are some common techniques:

1. Payback Period: This calculates how long it takes for the project's benefits to "pay back" the initial investment. It's easy to understand but doesn't consider the time value of money or ongoing benefits beyond the payback period.

2. Return on Investment (ROI): This expresses the project's profit as a percentage of the initial investment. It provides a broader picture than payback period but can be skewed by how you define benefits and costs.

3. Net Present Value (NPV): This takes into account the time value of money, discounting future benefits back to their present value. It's considered a more accurate analysis for long-term projects but requires more complex calculations.

4. Internal Rate of Return (IRR): This calculates the discount rate at which the NPV becomes zero. It essentially tells you the minimum acceptable return on investment for the project to be considered worthwhile.

5. Cost-Effectiveness Analysis: This focuses on comparing the costs of alternative options that achieve the same goal. It's helpful when different projects offer similar benefits but vary in cost.

6. Break-Even Analysis: This calculates the point where the project's costs and benefits equal each other. It helps you understand the minimum level of sales or usage needed to cover development costs.

Choosing the right technique:

The best technique depends on your specific project and needs. Consider factors like:

- **Project timeframe:** Payback period might be better for short-term projects, while NPV is more suitable for long-term ones.
- **Nature of benefits:** ROI works well for quantifiable benefits, while cost-effectiveness analysis is useful for comparing intangible benefits.
- **Your level of financial expertise:** Some techniques like NPV require more financial knowledge to interpret accurately.

Risk Evaluation

Risk evaluation in software project management is the essential process of **identifying, analyzing, and assessing potential risks** that could negatively impact your project's success. It's like looking at a weather forecast before starting a journey - you want to anticipate potential storms and be prepared to navigate them.

Here's how it works:

1. Risk Identification: This is where you brainstorm and list all the potential threats to your project, like technical challenges, budget overruns, or delays in resource availability.

2. Risk Analysis: Next, you delve deeper into each risk, estimating its **probability of occurring** and its **potential impact** on the project. For example, a security breach has a low probability but a high impact, while a minor bug might be more likely but cause less overall damage.

3. Risk Assessment: Based on the analysis, you categorize and prioritize the risks based on their severity. This helps you focus your attention and resources on the most critical threats.

4. Risk Mitigation: Finally, you develop strategies to **reduce the likelihood or impact** of each risk. This might involve contingency plans, proactive security measures, or additional training for your team.

Benefits of Risk Evaluation:

- **Improved project outcomes:** By proactively addressing risks, you increase your chances of staying on track and meeting your goals.
- **More informed decision-making:** Understanding potential risks helps you make better choices about resource allocation, scheduling, and project scope.
- **Enhanced communication and collaboration:** Openly discussing risks fosters transparency and helps team members work together to mitigate them.
- **Reduced stress and anxiety:** Knowing you've prepared for potential challenges can improve team morale and productivity.

Selection of Appropriate Report

Selecting the appropriate report in software project management depends on several factors, including:

1. Purpose of the report: Are you providing a status update, seeking funding approval, analyzing risks, or presenting results? Different purposes require different information and formats.

2. Target audience: Who will be reading the report? Tailor the content and complexity to their level of understanding and interest.

3. Project stage: Is the project in the planning, development, testing, or deployment phase? Each stage requires different information to be reported.

4. Project methodology: Are you using agile, waterfall, or another methodology? Different methodologies use different reporting formats and metrics.

5. Company standards: Does your company have specific reporting templates or guidelines? Ensure your report adheres to these.

Here are some common reports used in software project management and their typical uses:

- Project Status Report: Provides a high-level overview of project progress, milestones achieved, challenges faced, and upcoming tasks.

- Project Health Report: Dives deeper into project health aspects like budget adherence, schedule compliance, team morale, and risk mitigation measures.

- Project Risk Report: Identifies and analyzes potential risks, their likelihood and impact, and proposed mitigation strategies.

- Change Request Report: Documents proposed changes to the project scope, requirements, or timeline, along with justifications and potential impacts.

- Cost-Benefit Analysis Report: Analyzes the projected costs and benefits of a project to assess its financial viability.

- Test Summary Report: Summarizes the test results, identifies bugs or defects, and outlines their severity and resolution plans.

- Project Closure Report: Documents the project's completion, achievements, lessons learned, and final deliverables.

Notes By Anjali Kandel
NCMT college (Nepalgunj)

Project Approach

A **Project Approach** in software project management defines the overall set of principles, methodologies, and practices used to guide the planning, execution, and control of a software project. It's like choosing the blueprint and tools for building your software project.

Here are **key aspects of a Project Approach**:

1. Methodology: This identifies the overarching framework for managing the project, such as **waterfall, agile, Scrum, Kanban, Lean**, or a hybrid approach. Each methodology has its own lifecycle, deliverables, and emphasis on areas like collaboration and flexibility.

2. Practices: These are the specific techniques and tools used within the chosen methodology. For example, agile projects might use Kanban boards and daily stand-up meetings, while waterfall projects might rely on Gantt charts and formal change management processes.

3. Governance: This defines the decision-making structure, roles, and responsibilities within the project. It clarifies who makes key decisions, how stakeholders are involved, and how communication flows.

4. Customization: While methodologies offer frameworks, successful projects often adapt and tailor **the** approach to their specific needs, context, and team preferences.

Choosing the right approach:

The best project approach depends on various factors, including:

- **Project size and complexity:** Larger or more complex projects might benefit from structured methodologies like waterfall, while agile approaches might be better suited for smaller, rapidly evolving projects.
- **Team experience and preferences:** Consider your team's familiarity with different methodologies and their preferred work style.
- **Organizational culture and structure:** Align the approach with your company's overall project management practices and decision-making processes.
- **Project goals and priorities:** Tailor the approach to prioritize flexibility, speed, quality, or other specific project objectives.

Choosing Technologies

Choosing the right tech for your software project is like picking the perfect tools for a job Here's a quick guide:

1. **Know your project:** What are you building and who for?

2. **Consider your team:** Can they handle the tech smoothly?

3. **Security matters:** Does the tech meet your privacy and compliance needs?

4. **Think future-proof:** Will the tech scale with your growing project?

5. **Cost vs. benefit:** Is the price tag worth the features and performance?

6. **Do your research:** Learn from others and explore new options.

7. **Mix and match:** Try different techniques like prototypes and pilot projects.

8. **Be flexible:** Adapt your choices as your project and tech evolve.

Choice of Process Models

There are different project management methodologies that cater to the needs of different projects spanned across different business domains.

The goal of a software process model is to provide guidance for controlling and coordinating the tasks to achieve the end product and objectives as effectively as possible.

There are many different software processes but all involve:

1. **Specification**- defining what the system should do;
2. **Design and implementation** - defining the organization of the system and implementing the system;
3. **Validation** - checking that it does what the customer wants;
4. **Evolution**-changing the system in response to changing customer needs.

Choosing the right **process model** is crucial for setting the groundwork for success. It's like selecting a roadmap for your project, outlining the stages, activities, and decision

points you'll encounter. Each process model has its own strengths and weaknesses, and the best fit depends on your specific project needs and context.

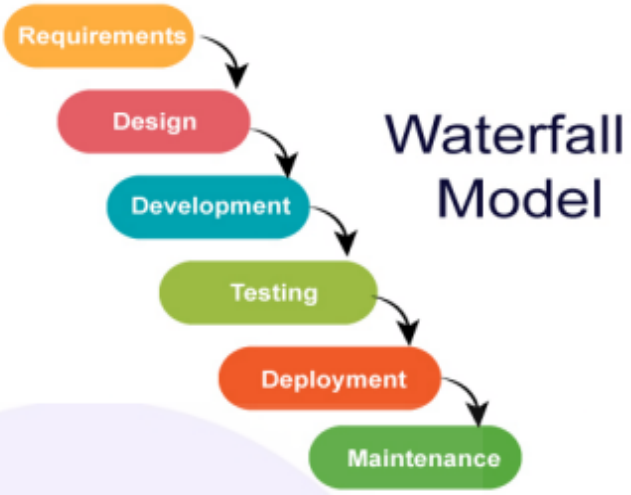
Different type of process model

1. Waterfall model
2. Incremental model
3. RAD model
4. Agile model
5. Iterative model
6. Spiral model
7. Prototype model

1. Waterfall Model

1. Plan-driven model.
2. Separate and distinct phases of specification and development
3. here are separate identified phases in the waterfall model:
    I. Requirement's analysis and definition
    II. System and software design
    III. Implementation and unit testing
    IV. Integration and system testing
    V. Operation and maintenance
4. The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

5. When to Use It: The Waterfall approach is great for manufacturing and construction projects, which are highly structured.



2. Incremental Model

An incremental approach breaks the software development process down into small, manageable portions known as increments. Each component delivered must give some benefit to the user. Above figure of incremental delivery Great for projects that have loosely-coupled parts and projects with complete and clear requirements

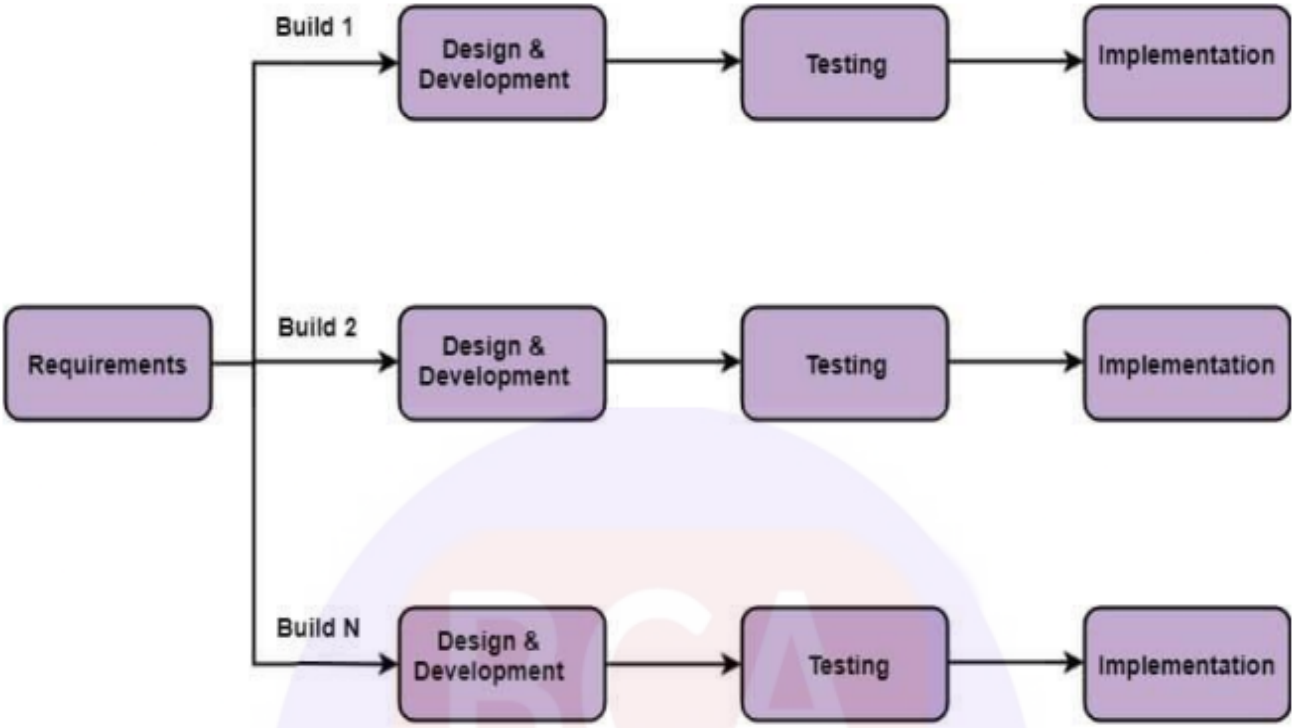


Fig: Incremental Model

3. RAD (Rapid Application Development) Model

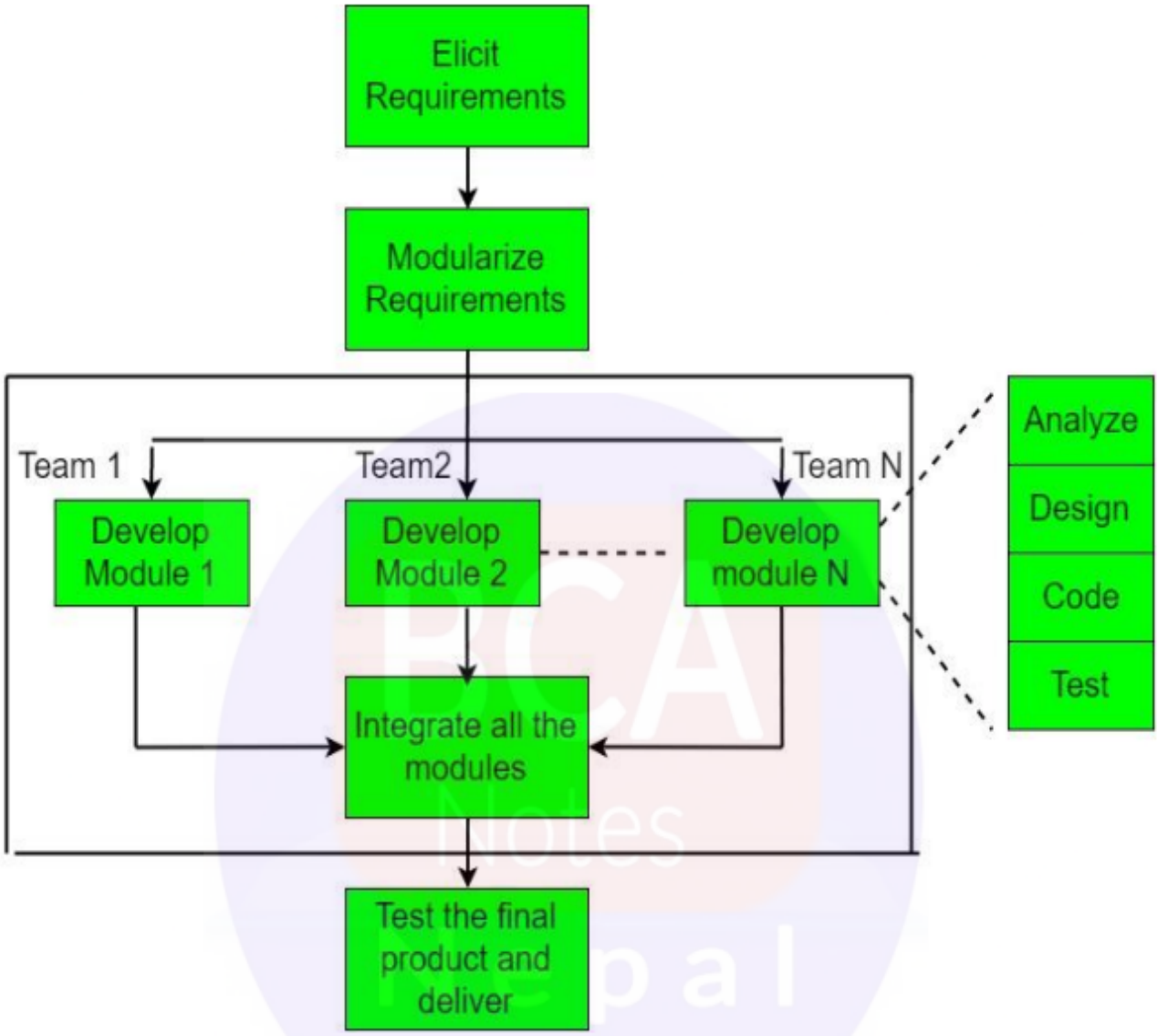
The Rapid Application Development (RAD) model in software project management is like building a house one room at a time:

Key features:

- **Fast and iterative:** Develop in short cycles, getting feedback after each "room" (prototype).
- **User-focused:** Involve users early and often to shape the final product.
- **Flexible:** Easy to adapt to changing needs as you build.

Benefits:

- **Faster development:** Get your software to market quicker.
- **Happier users:** They help shape the product they'll use.
- **Better quality:** Catch issues early through prototypes.



Notes By Anjali Kandel
NCMT college (Nepalgunj)

4. Agile Model

These methods:

1. Focus on the code rather than the design
2. Are based on an iterative approach to software development
3. Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.
4. The aim of agile methods is to reduce overheads in the software process (e.g., by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.

Notes By Anjali Kandel
NCMT college (Nepalgunj)

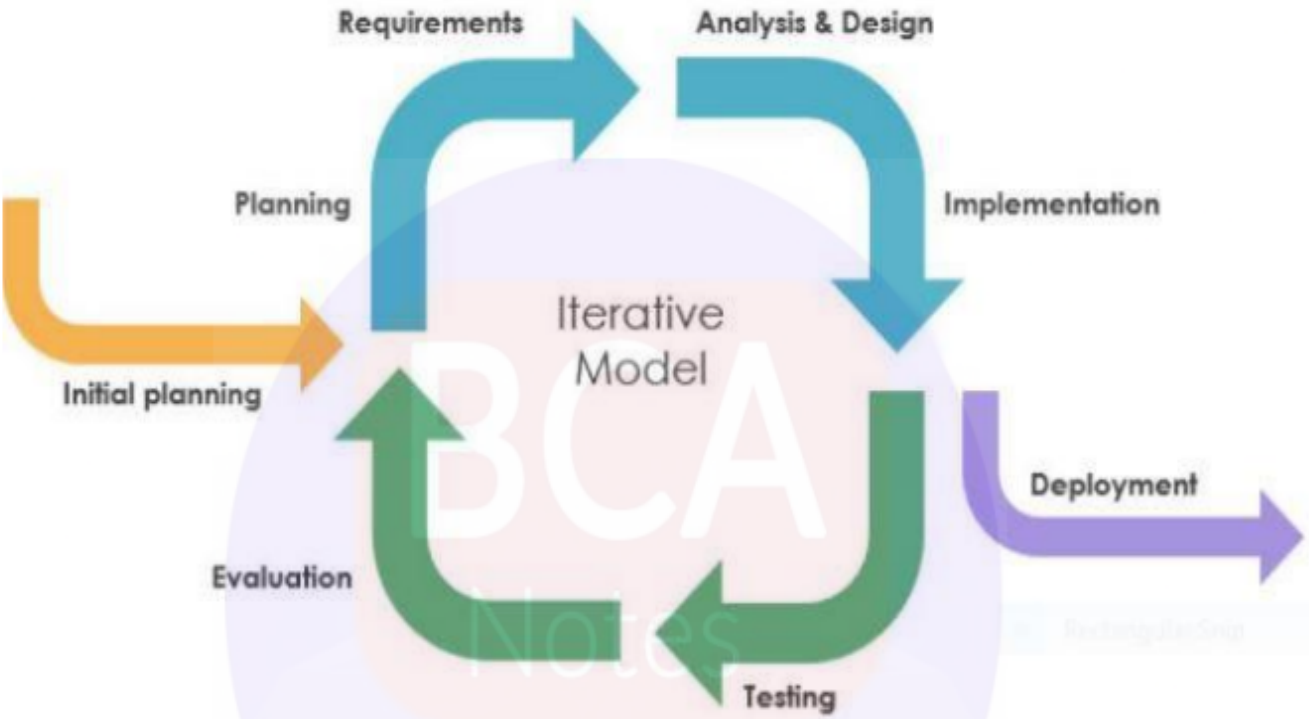
The Principle Of Agile Methods

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Notes By Anjali Kandel
NCMT college (Nepalgunj)

5. Iterative Model

The iterative process model is the implementation of the software development life cycle in which the initial development is started based on the initial requirements and more features are added to the base software product with the ongoing iterations until the final system is created.



6. Spiral Model

1. Process is represented as a spiral rather than as a sequence of activities with backtracking.
2. Each loop in the spiral represents a phase in the process.
3. No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
4. Risks are explicitly assessed and resolved throughout the process.

Advantages

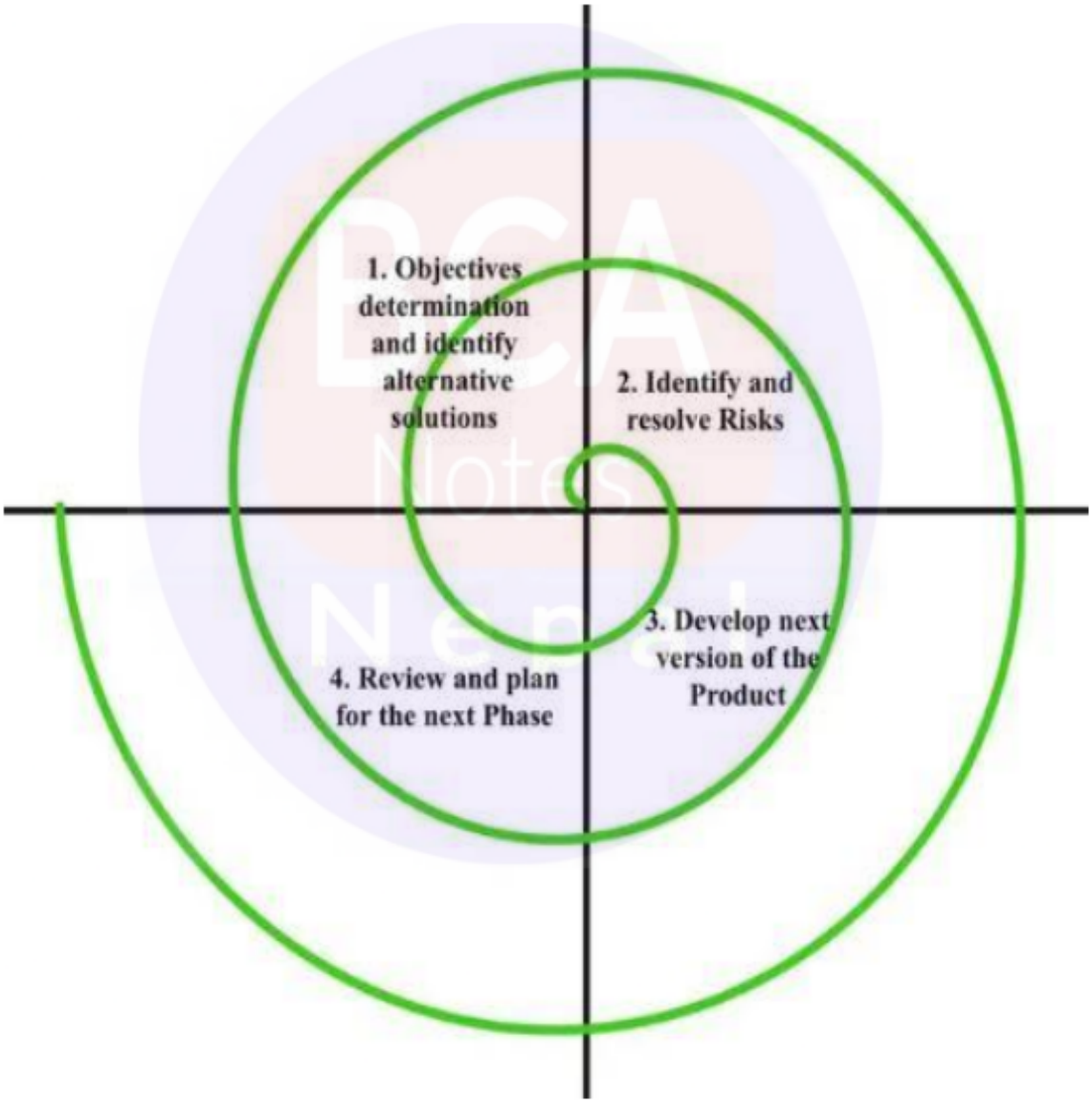
1. Changing requirements can be accommodated.
2. Allows extensive use of prototypes.
3. Requirements can be captured more accurately.

Notes By Anjali Kandel
NCMT college (Nepalgunj)

4. Users see the system early.

Disadvantage

1. Management is more complex.
2. End of the project may not be known early.
3. Not suitable for small or low risk projects and could be expensive for small projects.
4. Process is complex



7. Prototype Model

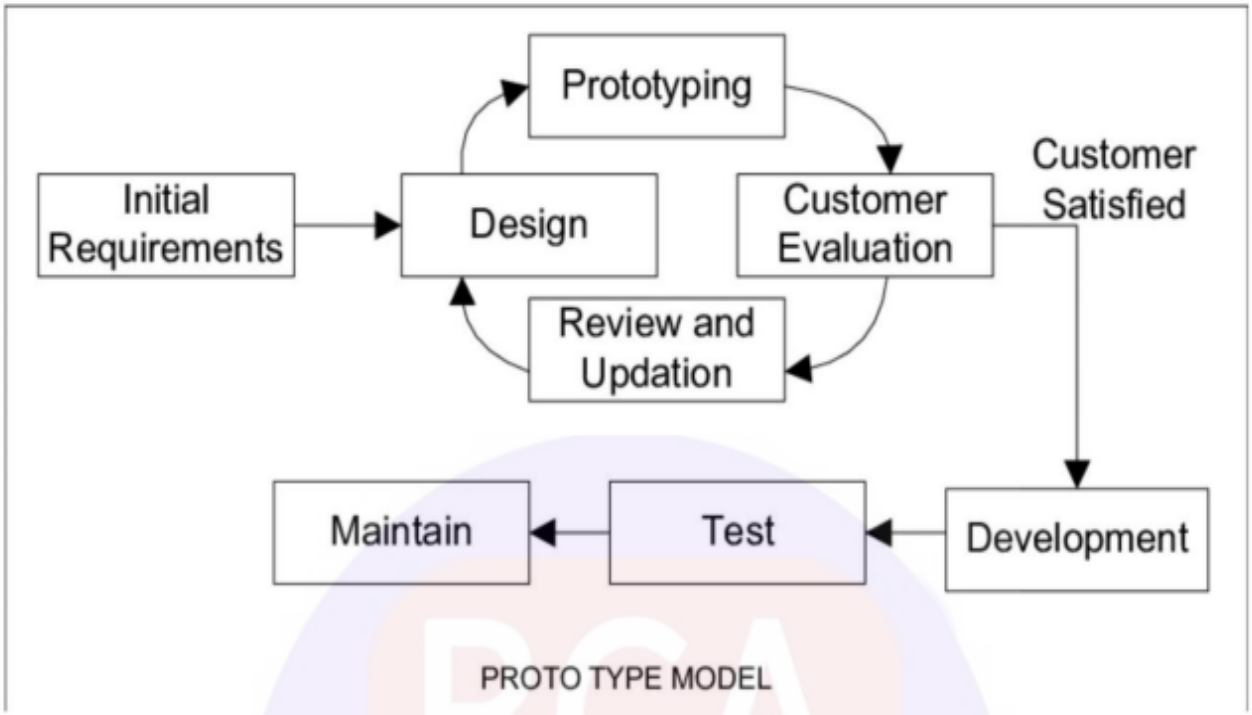
1. A prototype is a working model of one or more aspects of the projected system.
2. It is constructed and tested quickly and inexpensively in order to test out assumptions.
3. Because Uncertainty about a project is usually because of a lack of knowledge about some aspect So, prototyping is the need to reduce uncertainty by conducting an experiment.
4. The most important reason for prototyping is a need to learn about an area of uncertainty.
5. Thus, it is essential to identify at the outset what is to be learnt from the prototype.

Here are the key **characteristics** of the prototype model:

- **Focus on prototypes:** You create multiple prototypes throughout the development lifecycle, starting with low-fidelity prototypes (basic mockups) and progressing to high-fidelity prototypes (more advanced simulations) as you gather feedback and refine the design.
- **Iterative development:** Each prototype serves as a learning experience, guiding the development of the next iteration, with the final product gradually taking shape through these iterations.
- **Early user involvement:** Stakeholders and users are actively involved in evaluating and providing feedback on prototypes, ensuring the final product meets their needs and expectations.
- **Flexibility:** The model allows for adapting requirements and functionalities based on feedback and emerging needs, making it ideal for projects with uncertain or evolving requirements.

Benefits/Advantages of the prototype model:

- **Reduced risks:** Identifying and addressing issues early through prototypes minimizes the risk of major errors in the final product.
- **Improved user satisfaction:** Early user involvement ensures the final product aligns with their needs and expectations, leading to higher satisfaction.
- **Increased communication and collaboration:** The model fosters communication and collaboration between developers, stakeholders, and users, promoting project clarity and alignment.
- **Flexibility and adaptability:** It allow for changes in requirements as the project progresses, making it suitable for dynamic environments.



PROTO TYPE MODEL

Structured Methods

Structured methods consist of sets of steps and rules which, when applied, generate system products such as use case diagrams. Each of these products is carefully defined. Such methods are more time consuming and expensive than more intuitive approaches. Structured methods refer to a collection of formal, step-by-step approaches used to develop and manage software projects. These methods aim to bring control, organization, and consistency to the development process, ensuring projects are delivered on time, within budget, and meet specified requirements.

A structured method includes a design process model, notations to represent the design, report formats, rules and design guidelines. Structured methods may support some or all of the following models of a system:

1) An object model that shows the object classes used in the system and their dependencies.
2) A sequence model that shows how objects in the system interact when the system is executing.
3) A state transition model that shows system states and the triggers for the transitions from one state to another.

4) A structural model where the system components and their aggregations are documented.
5) A data flow model where the system is modelled using the data transformations that take place as it is processed. This is not normally used in object-oriented methods but is still frequently used in real-time and business system design.
6) A use-case model that shows the interactions between users and the system.

Key characteristics of structured methods:

- **Formal and well-defined:** They define clear phases, activities, and deliverables for each stage of the development process.
- **Emphasis on documentation:** Extensive documentation is created throughout the project, capturing requirements, design decisions, and test results.
- **Structured tools and techniques:** Specific tools and techniques are recommended for various tasks, like data flow diagrams for data analysis and entity-relationship diagrams for modeling data relationships.
- **Focus on quality control:** Quality assurance activities are integrated throughout the development process to identify and address issues early on.

Different types of structured methods:

- **Waterfall:** A sequential approach where each phase (requirements, design, development, testing, deployment) is completed before moving on to the next.
- **Spiral:** Combines features of waterfall and iterative models, allowing for risk assessment and prototyping before full commitment to each phase.
- **Prototyping:** Develops simplified versions of the software to gather user feedback and refine requirements before full development.
- **Agile:** Employs short iterations (sprints) with continuous delivery and feedback loops, adapting to changing requirements as the project progresses.

Benefits of using structured methods:

- **Reduced risk:** Formal processes help identify and mitigate potential problems early on.
- **Improved project control:** Clear phases and deliverables aid in tracking progress and managing resources effectively.
- **Enhanced communication:** Detailed documentation facilitates communication between stakeholders and team members.
- **Better quality:** Emphasis on quality control ensures higher quality software is delivered.

Challenges of using structured methods:

- **Can be inflexible:** May not adapt well to rapid changes in requirements or technology.
- **Time-consuming:** Extensive documentation can be resource-intensive and time-consuming.
- **Bureaucratic:** Strict adherence to processes might hinder creativity and innovation.

Notes By Anjali Kandel
NCMT college (Nepalgunj)