ENGINEERING ENTREPRENEURSHIP AND IPR

Module 3

- **3.1 Business plan preparation**
- **3.2 Prototype development plan preparation**

Module 3 Syllabus Contd…

Prototype development plan preparation

- Prototype requirements analysis
- Technical specifications
- Development approach
- Development timeline
- Resource allocation
- Testing and quality assurance
- Iterative development and feedback loop
- Documentation and version control

INTRODUCTION

- To launch a business the following are needed.
  - Business Plan
  - Prototype

Prototype Development Plan

- A prototype development plan is a strategic roadmap that outlines the key activities, timelines, and resources required to design, build, and test a prototype.
- It's a crucial document that helps stakeholders visualize the project's scope, milestones, and expected outcomes.
- The plan typically includes the steps, resources, and timelines necessary to transform an idea into a functional prototype.

Key Components of Prototype Development Plan Preparation

- Objective Definition
- Scope and Requirements
- Resource Allocation
- Timeline and Milestones
- Design and Development Plan
- Testing and Evaluation
- Documentation
- Risk Management

Key Components of Prototype Development Plan Preparation

- **Objective Definition**
- Clearly define the goals and purpose of the prototype.
- What specific problem does it aim to solve or what functionality does it need to demonstrate?
- **Scope and Requirements**
- Determine the scope of the prototype, including the features and functionalities it will include.
- Identify the technical and material requirements needed for development.

Key Components of Prototype Development Plan Preparation

Resource Allocation: Plan for the necessary resources, including materials, tools, technology, and personnel. This step ensures that everything required for the prototype is available.

Timeline and Milestones: Establish a timeline with specific milestones to track progress. This helps in managing the project efficiently and ensures timely completion.

Key Components of Prototype Development Plan Preparation

Design and Development Plan: Outline the design process, including initial sketches, CAD models, and detailed drawings. Plan the development stages, from initial concept to final prototype.

Testing and Evaluation: Plan for testing the prototype to evaluate its performance and identify any issues. This includes setting criteria for success and methods for gathering feedback.

Key Components of Prototype Development Plan Preparation

Documentation: Keep detailed records of the entire process, including design changes, test results, and feedback. This documentation is crucial for future reference and iterations.

Risk Management: Identify potential risks and challenges that might arise during the development process and plan for mitigation strategies.

Key Components of Prototype Development Plan Preparation

By following these steps, the prototype development plan preparation ensures a structured and efficient approach to creating a prototype.

A well-prepared prototype development plan ensures a smooth and efficient project execution, minimizing the risk of delays, cost overruns, and product failures.

Prototype Requirement Analysis

- Prototype requirements analysis is a process that helps organizations understand customer requirements and develop a prototype that meets those needs.
- It involves identifying and documenting the specific needs and constraints that the prototype must meet.
- This process ensures that theprototypewill be functional, feasible, and aligned with the project's goals.

Prototype Requirement Analysis – Key Steps

1. Gathering Requirements: Collecting detailed information from stakeholders about what the prototype should achieve. This includes functional requirements (what the prototype should do) and non-functional requirements (how the prototype should perform).

2. Analysing Requirements: Evaluating the gathered requirements to ensure they are clear, complete, and feasible. This step often involves prioritizing requirements based on their importance and impact.

3. Documenting Requirements: Creating detailed documentation that outlines all the requirements. This serves as a reference throughout the development process and helps ensure that all stakeholders have a shared understanding of the project's goals.

4. Validating Requirements: Confirming that the requirements accurately reflect the needs of the stakeholders and are achievable within the project’s constraints. This may involve reviewing the requirements with stakeholders and making necessary adjustments.

Example - Development of a Low-Cost Medical Device

- **Gathering Requirements**: For a project aimed at developing a low-cost medical device for rural healthcare, requirements might include affordability, ease of use, portability, and reliability. Stakeholders such as healthcare professionals, patients, and regulatory bodies would provide input.

- **Analyzing Requirements**: The team would analyse these requirements to ensure they are feasible given the budget and technological constraints. For instance, they might prioritize features that are essential for basic healthcare over advanced functionalities.

Example - Development of a Low-Cost Medical Device

- **Documenting Requirements:** Detailed documentation would be created specifying the device's functionalities such as measuring vital signs, and non-functional requirements like battery life and durability.
- **Validating Requirements:** The requirements would be validated through discussions with healthcare providers and pilot testing in rural clinics to ensure the device meets the needs of the target users.

Example - Smart Agriculture Solution

Gathering Requirements: For a smart agriculture project, requirements might include real-time soil monitoring, weather forecasting, and automated irrigation.

- Input wouldbe gathered from farmers, agricultural experts, and technology providers.
- **Analysing Requirements**: The team would analyse these requirements to balance cost, technological feasibility, and user-friendliness.
- For example, they might decide to focus on essential features like soil moisture sensors and basic weather alerts.

Example - Smart Agriculture Solution

- **Documenting Requirements**: The documentation would detail the system's functionalities, such as data collection and analysis, and non-functional requirements like system reliability and ease of installation.
- **Validating Requirements**: The requirements would be validated through field trials and feedback from farmers to ensure the solution is practical and beneficial for the agricultural community.

Technical Specifications

- **Technical Specifications** in the context of prototype development plan preparation refer to detailed descriptions of the technical requirements and standards that the prototype must meet.
- They typically include information on materials, dimensions, performance criteria, and testing methods.
- These specifications guide the development process, ensuring that the prototype functions correctly and meets the project's goals.

Technical Specifications

- Technical specifications declare how the functional specifications will be accomplished.
- Technical specifications are different from functional specifications, which declare what a product needs to do, how it behaves, and how it looks.

Key Components of Technical Specifications

1. **Functional Requirements**: What the prototype should do.
2. **Performance Requirements**: How well the prototype should perform.
3. **Design Specifications**: Detailed drawings, dimensions, and materials.
4. **Compliance Standards**: Relevant industry standards and regulations.
5. **Testing Procedures**: Methods for verifying that the prototype meets the specifications.

Technical Specifications for Solar-Powered Water Pump for Agriculture

- **Functional Requirements**: The pump should be able to draw water from a depth of up to 30 meters and deliver it to the surface for irrigation purposes.
- **Performance Requirements**: It should operate efficiently with a solar power input of 300 watts and provide a flow rate of at least 20 liters per minute.
- **Design Specifications**: The pump should be made of corrosion-resistant materials suitable for rural environments, with a maximum weight of 15 kg for ease of transport and installation.
- **Compliance Standards**: Must comply with the Bureau of Indian Standards (BIS) for agricultural pumps.
- **Testing Procedures**: Field tests to ensure the pump operates effectively under various weather conditions and maintains performance over a specified period.

Technical Specifications for Affordable Electric Scooter for Urban Commuting

- **Functional Requirements**: The scooter should have a range of at least 60 kilometers on a single charge and a top speed of 50 km/h.
- **Performance Requirements**: It should be able to handle inclines of up to 10 degrees and support a maximum load of 150 kg.
- **Design Specifications**: The scooter should have a lightweight frame made from durable materials, with a total weight not exceeding 100 kg. It should also include safety features like LED lights and a reliable braking system.
- **Compliance Standards**: Must meet the Automotive Industry Standards (AIS) for electric vehicles in India.
- **Testing Procedures**: Performance tests on various road conditions, battery life tests, and safety tests to ensure compliance with AIS standards.

Development Approach

- Development Approach in the context of prototype development plan preparation refers to the strategy and methodology used to create a prototype.
- This approach outlines the steps, techniques, and processes that will be followed to ensure the prototype meets its intended goals. It includes decisions on design, materials, testing, and iteration.

Development approaches can include:

- **Predictive**: A traditional approach that follows a set of steps, such as requirements, design, implementation, testing, deployment, and maintenance
- **Iterative**: A method that involves repeated cycles of activities
- **Incremental**: A method that involves making small, incremental changes
- **Agile**: A method that's used in software development
- **Hybrid**: A method that blends elements of multiple approaches

- The goal of a development approach is to maximize the value of a project while achieving quality, time, and cost standards

Development Approach – Key Components

1. **Methodology**: Choosing betweenmethodologies depending on the project's needs.
2. **Design Process**: Steps for conceptualizing, designing, and refining the prototype.
3. **Material Selection**: Deciding on the materials and components to be used.
4. **Testing and Validation**: Methods for testing the prototype to ensure it meets requirements.
5. **Iteration and Feedback**: Processes for incorporating feedback and making improvements.

Development Approach – Key Components - Development of a Smart Irrigation System

- **Methodology**: An Agile approach is chosen to allow for flexibility and iterative improvements based on farmer feedback.
- **Design Process**: Initial designs are created using CAD software, followed by the development of a basic prototype with sensors and control systems.
- **Material Selection**: Durable, weather-resistant materials are selected to withstand harsh agricultural environments.
- **Testing and Validation**: Field tests are conducted in various climatic conditions to ensure the system's reliability and efficiency.
- **Iteration and Feedback**: Feedback from farmers is gathered and used to make iterative improvements, such as enhancing the user interface for easier operation.

Development Approach – Key Components - Development of a Low-Cost Portable Water Purifier

- **Methodology**: A Lean approach is adopted to minimize waste and focus on essential features.
- **Design Process**: The design process includes creating detailed sketches and 3D models, followed by the construction of a prototype using locally available materials.
- **Material Selection**: Affordable and readily available materials are chosen to keep costs low while ensuring effectiveness.
- **Testing and Validation**: The prototype undergoes rigorous testing for water purification efficiency and durability.
- **Iteration and Feedback**: User feedback from rural communities is collected to refine the design, ensuring it meets the needs of the target population.

Key Factors Influencing the Timeline

- **Complexity of the Design**
  The level of detail and functionality required in the prototype significantly affects the timeline. Simple designs may require minimal time, whereas high-fidelity prototypes demand extended development periods.
- **Resource Availability**
  Access to materials, tools, and skilled personnel can either accelerate or delay the development process.
- **Technological Tools**
  The use of advanced technologies, such as 3D printing, computer-aided design (CAD), or CNC machining, can streamline processes and reduce the time required for development.

Key Factors Influencing the Timeline contd…

- **Iteration Requirements**
  Prototyping is inherently iterative. The number of cycles needed to refine the design based on feedback can extend the timeline.
- **Team Collaboration**
  Effective communication and collaboration among designers, engineers, and stakeholders ensure timely progress.
- **Regulatory and Testing Requirements**
  Prototypes intended for industries like healthcare or aerospace may need to adhere to stringent standards, increasing the development timeline.

Resource Allocation

- **Resource Allocation** refers to the strategic distribution and management of resources such as time, budget, personnel, materials, and equipment necessary to develop a prototype.
- **It involves:**
  - Identifying the resources needed to complete the project, such as team members, tools, and budget.
  - Determining how to use the resources most effectively and economically.
  - Assigning the resources to tasks or activities within the project.

Resource Allocation - Tips

- Think through the constraints, resources, and how to position each team member.
- Prioritize the activities that will move the business forward.
- Be mindful of the main initiatives, constraints, and focuses.

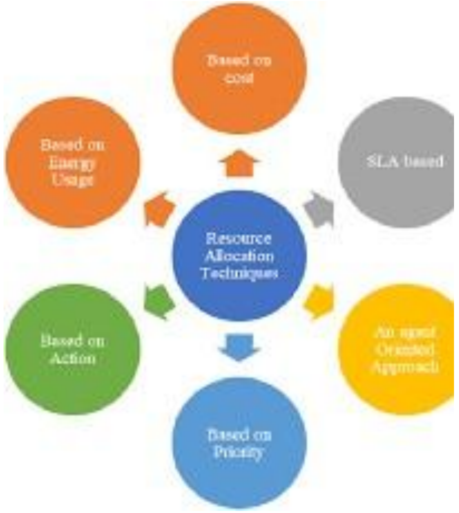
Resource Allocation – Key Components

1. **Identifying Resources**: <u>Determining what resources are needed for the project</u>, including human resources (engineers, designers, technicians), materials, tools, and technology.

2. **Budgeting**: <u>Allocating financial resources to different aspects</u> of the project, ensuring that each phase has the necessary funding.

3. **Scheduling**: <u>Planning the timeline for resource usage, ensuring that resources are available when needed</u> and avoiding bottlenecks.

4. **Monitoring and Adjusting**: Continuously <u>tracking resource usage and making adjustments</u> as necessary to address any issues or changes in the project scope.



Resource Allocation – Example - Development of a Rural Electrification Prototype

- **Identifying Resources**: The project requires solar panels, batteries, inverters, and skilled technicians. Additionally, community engagement specialists are needed to educate local residents.
- **Budgeting**: Financial resources are allocated to purchase solar panels and batteries, with a portion set aside for training and community outreach programs.
- **Scheduling**: The project timeline includes phases for procurement, installation, and testing, ensuring that technicians and materials are available at each stage.
- **Monitoring and Adjusting**: Regular progress reviews are conducted to ensure that the project stays on track. If delays occur, additional resources or adjustments to the schedule are made to keep the project moving forward.

Resource Allocation – Example - Development of a Low-Cost Educational Tablet

- **Identifying Resources**: The project needs hardware components (screens, processors, batteries), software developers, and educational content creators.
- **Budgeting**: Funds are allocated for hardware procurement, software development, and content creation, with a contingency fund for unexpected expenses.
- **Scheduling**: A detailed timeline is created, outlining the stages of hardware assembly, software integration, and content development, ensuring that each team has the resources they need at the right time.
- **Monitoring and Adjusting**: The project manager regularly review the resource usage and progress. If the software development is ahead of schedule, resources might be reallocated to speed up hardware assembly or content creation.

Testing and Quality Assurance

- **Testing and Quality Assurance (QA)** in the context of prototype development plan preparation involves systematically evaluating the prototype to ensure it meets the specified requirements and standards.

- This process helps identify defects, ensure functionality, and verify that the prototype performs as expected under various conditions.

Testing and Quality Assurance – Key Components

1. **Functional Testing**: Verifying that the prototype performs its intended functions correctly.
2. **Performance Testing**: Assessing how well the prototype performs under different conditions, such as stress or load.
3. **Usability Testing**: Ensuring the prototype is user-friendly and meets the needs of its intended users.
4. **Compliance Testing**: Checking that the prototype adheres to relevant standards and regulations.
5. **Iterative Testing**: Continuously testing and refining the prototype based on feedback and test results

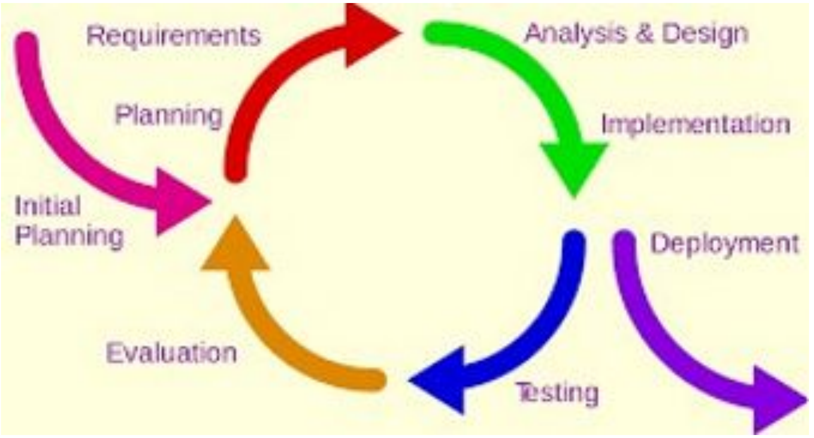
Testing and Quality Assurance – Examples - Development of a Smart Water Management System

- **Functional Testing**: The system is tested to ensure it can accurately monitor water levels, detect leaks, and control water distribution. Sensors and control units are evaluated for accuracy and reliability.
- **Performance Testing**: The system is subjected to various stress tests, such as simulating high water demand periods and potential system failures, to ensure it can handle peak loads and recover from faults.
- **Usability Testing**: Farmers and local water management authorities are involved in testing the system to ensure it is easy to use and provides valuable insights. Feedback is gathered to improve the user interface and functionality.
- **Compliance Testing**: The system is checked against local and national water management regulations to ensure it meets all legal requirements.

Testing and Quality Assurance – Examples - Development of a Low- Cost Electric Vehicle (EV) for Urban Areas

- **Functional Testing**: The EV is tested to ensure all components, such as the motor, battery, and control systems, work correctly. Safety features like brakes and lights are also tested.
- **Performance Testing**: The EV undergoes tests to evaluate its range, speed, and battery life under different driving conditions, such as city traffic and highway speeds.
- **Usability Testing**: Potential users, including commuters and delivery drivers, test the EV to ensure it is comfortable, easy to drive, and meets their needs. Feedback is used to improve the design and features.
- **Compliance Testing**: The EV is tested to ensure it complies with Indian automotive standards and regulations, including safety and environmental requirements.
- **Iterative Testing**: Based on the feedback and test results, the system is refined and retested to address any issues and improve performance.

Iterative Development and Feedback loop



- **Iterative development and feedback loop** is a process in prototype development where the prototype is continuously improved through repeated cycles of development, testing, and feedback.
- This approach allows for incremental enhancements and ensures that the final product closely aligns with user needs and expectations.

Iterative development and feedback loop

Iterative development and feedback loops can help improve the development process in several ways:

- **Faster feedback loops**: Teams can identify issues early and adjust quickly.
- **Reduced project failure risk**: Teams can validate features during development and adjust their strategy quickly if something isn't working.
- **Improved user experience**: Teams can collect customer feedback and use it to improve the product.
- **Increased agility**: Teams can adapt to changing needs and technology with ease.
- **Improved communication**: Regular feedback loops promote open communication among developers, designers, and stakeholders.

Iterative development and feedback loop – Key Components

1. **Iteration**: Developing the prototype in small, manageable increments rather than all at once. Each iteration builds on the previous one, incorporating improvements and new features.

2. **Feedback Collection**: Gathering feedback from users, stakeholders, and testing results after each iteration. This feedback is crucial for identifying issues and areas for improvement.

3. **Refinement**: Using the feedback to make necessary adjustments and improvements to the prototype. This cycle repeats until the prototype meets the desired standards and requirements.

Iterative development and feedback loop – Example- Development of a Mobile Health App for Rural Areas

- **Iteration**: The initial version of the app includes basic features like appointment scheduling and health tips. Subsequent iterations add more advanced features such as telemedicine consultations and health monitoring.
- **Feedback Collection**: Feedback is gathered from rural healthcare workers and patients using surveys and focus groups. They provide insights into usability, functionality, and any technical issues.
- **Refinement**: Based on the feedback, the app is refined to improve user interface design, add local language support, and enhance connectivity in low-bandwidth areas. This process continues until the app is fully optimized for its users.

Iterative development and feedback loop – Example- Development of a Smart Classroom Solution

- **Iteration**: The first iteration includes basic functionalities like digital attendance and content sharing. Later iterations introduce interactive features such as virtual labs and real-time quizzes.
- **Feedback Collection**: Teachers and students provide feedback through regular testing sessions and workshops. Their input helps identify usability issues and additional features needed.
- **Refinement**: The development team uses this feedback to refine the system, making it more user-friendly and adding features like offline access and integration with existing educational tools. This iterative process ensures the solution effectively meets the needs of Indian classrooms.

Document Version Control

- **Documentation and version control** are essential components of the prototype development plan preparation. They ensure that all project information is accurately recorded, easily accessible, and systematically managed throughout the development process.

.

Document Version Control

- **Documentation -** Documentation involves creating detailed records of all aspects of the prototype development process. This includes design specifications, test results, user feedback, and any changes made during development. Proper documentation helps maintain clarity, facilitates communication among team members, and provides a reference for future projects.
- **Version Control -** Version control is a system that tracks changes to documents and files over time. It allows developers to manage multiple versions of a document, ensuring that all changes are recorded and previous versions can be retrieved if needed. This is crucial for maintaining the integrity of the project and avoiding confusion caused by multiple team members working on the same files.

Document Version Control – Key Components

1. **Change Tracking**: Recording all modifications made to documents, including who made the changes and when.
2. **Version History**: Keeping a history of all document versions, allowing for easy comparison and rollback if necessary.
3. **Access Control**: Managing who can view or edit documents to ensure that only authorized personnel make changes.
4. **Collaboration**: Facilitating teamwork by allowing multiple users to work on documents simultaneously while keeping track of individual contributions.

Document Version Control – Example - Development of a Smart City Infrastructure Prototype

- **Documentation**: <u>Detailed records are kept for each phase of the project, including initial designs, stakeholder feedback, and test results.</u> This documentation helps ensure that all aspects of the project are transparent and traceable.
- **Version Control**: A version control system is used to manage changes to design documents and software code. For example, <u>if a new traffic management algorithm is developed, all changes are tracked, and previous versions can be accessed.</u>

Document Version Control – Example - Development of a Low-Cost Sanitation Solution

- **Documentation**: The project team documents the design process, materials used, and testing procedures. This includes detailed reports on the effectiveness of different materials and designs in various environmental conditions.
- **Version Control**: Version control is applied to design blueprints and testing protocols. If a new material is tested for durability, the results are documented, and the version control system tracks all changes to the testing procedures and results. This ensures that the team can revert to previous methods if the new material does not perform as expected.

Document Version Control – Benefits

Improved Collaboration:

- **Multiple Contributors**: Allows multiple people to work on the same document without overwriting each other's changes.
- **Real-Time Updates**: Teams can see updates and changes in real-time, enhancing collaborative efforts.

Historical Tracking:

- **Change History**: Keeps a detailed history of all changes made to a document, including who made each change and when.
- **Revert to Previous Versions**: Facilitates reverting to earlier versions if a mistake is made, saving time and effort.

Enhanced Accountability:

- **Traceability**: Each version is attributed to its editor, making it easy to track accountability.
- **Audit Trail**: Provides an audit trail for compliance and regulatory purposes.

Document Version Control – Benefits

Conflict Resolution:

- **Merge Conflicts**: Helps in identifying and resolving conflicts when multiple people edit the same document simultaneously.
- **ClearComparisons**: Enables easycomparison of different versions to identify and reconcile differences.

Improved Organization:

- **Structured Management**: Organizes documents and their versions in a structured
manner, reducing clutter and confusion.
- **Consistent Naming**: Maintains consistent version naming and numbering conventions.

Data Integrity and Security:

- **Backup**: Automatically backs up previous versions, protecting against data loss.
- **Controlled Access**: Limits access to certain versions, ensuring sensitive information is protected.

Document Version Control – Limitations

Complexity:

- **Learning Curve**: Requires users to learn and adapt to version control systems, which can be complex for beginners.
- **Implementation**: Setting up and managing version control systems can be time-consuming and requires technical expertise.

Storage and Performance:

- **Storage Requirements**: Storing multiple versions can require significant storage space.
- **System Performance**: Large version histories can affect the performance of the version control system.

Overhead:

- **Administrative Overhead**: Managing versions, resolving conflicts, and maintaining version control systems can add administrative overhead.
- **Manual Effort**: Sometimes requires manual intervention to merge changes or resolve conflicts, increasing the workload.

Document Version Control – Limitations

Dependence on Technology:

- **System Reliance**: Highly reliant on the version control system and its availability. System failures can disrupt work.
- **Software Costs**: Depending on the version control system used, there may be costs associated with licensing and maintenance.

Potential for Errors:

- **Human Error**: Mistakes can still occur if users fail to follow version control protocols correctly.
- **Version Confusion**: Without proper training, users might get confused between different versions, leading to errors.

THANK YOU