1. **Linear Project Management Framework:**

1.1 Overview of Project Management: Project management involves the application of knowledge, skills, tools, and techniques to meet project requirements. It includes initiating, planning, executing, controlling, and closing activities to achieve project objectives.

1.2 Project Management Life Cycle - IEEE Life Cycle: The IEEE (Institute of Electrical and Electronics Engineers) defines a standard project life cycle that consists of phases like conception, initiation, planning, execution, control, and closure. These phases provide a structured approach for managing software projects.

1.3 Project Management Process: The project management process comprises a set of interrelated activities that are performed to achieve project objectives. It typically includes processes like project initiation, project planning, project execution, project monitoring and control, and project closure.

1.4 Role of Project Manager: The project manager is responsible for planning, executing, and closing projects. Their role involves defining project objectives, managing project resources, coordinating team members, monitoring progress, managing risks, and ensuring project success.

1.5 Quality Metrics: Quality metrics are measurements used to evaluate the quality of software products or processes. These metrics help assess factors like code complexity, defect density, test coverage, customer satisfaction, and adherence to standards. They provide insights into the overall quality of the project and aid in making informed decisions.

1.6 Risk Management Process (Case Study Based): 1.6.1 Risk Identification: Risk identification involves identifying potential risks that may impact the project. This can be done through techniques like brainstorming, checklists, and historical data analysis. Risks can be categorized as technical, organizational, or external.

1.6.2 Risk Analysis: Risk analysis assesses the probability and impact of identified risks. Qualitative and quantitative techniques are used to prioritize risks based on their severity. This helps in determining the appropriate risk response strategies.

1.6.3 Risk Mitigation: Risk mitigation involves developing strategies to reduce the probability or impact of identified risks. Strategies may include contingency planning, risk transfer, risk avoidance, or risk acceptance. The goal is to minimize the potential negative impact on the project.

1.6.4 RMMM (Risk Mitigation, Monitoring, and Management): RMMM refers to a plan developed to address identified risks. It includes specific actions to mitigate risks, monitoring mechanisms to track risks throughout the project, and management approaches to ensure effective risk response and control.

1.7 Hands-on MS Project Tool - Resource Allocation, Scheduling, Gantt Chart: Microsoft Project is a popular project management tool that helps in resource allocation, scheduling, and visualizing project timelines through Gantt charts. It allows project managers to assign tasks, allocate resources, track progress, and generate reports to manage projects effectively.

2. **Linear Software Project Estimation:**

2.1 Different Methods of Cost Estimation: 2.1.1 COCOMO-I & II Model (Problem Statement): COCOMO (Constructive Cost Model) is a software cost estimation model. COCOMO-I is a basic model used for estimating small projects, while COCOMO-II is an improved model that considers various factors like project size, complexity, and team experience to estimate project effort and duration.

2.1.2 Delphi Cost Estimation: Delphi estimation is a technique used to estimate project costs by involving multiple experts. It involves collecting individual estimates, summarizing them, and iteratively refining the estimates until a consensus is reached. This approach helps in reducing bias and improving estimation accuracy.

2.2 Function Point Analysis (Problem Statement): Function Point Analysis (FPA) is a technique for estimating the size and complexity of a software system. It measures the functionality provided by the system and assigns function points based on various components like inputs, outputs, inquiries, files, and interfaces. FPA helps in estimating effort, cost, and duration of a project.

2.3 The SEI Capability Maturity Model (CMM): The Capability Maturity Model (CMM) is a framework that defines the maturity levels of an organization's software development processes. It provides a roadmap for process improvement and helps organizations assess their capabilities and identify areas for improvement.

2.4 Software Configuration Management: Software Configuration Management (SCM) is the process of managing and controlling changes to software products throughout their lifecycle. It includes version control, build management, release management, and change management. SCM ensures that software is delivered in a controlled and consistent manner.

3. **Agile Project Management Framework:**

3.1 Introduction and Definition Agile, Agile Project Life Cycle: Agile project management is an iterative and incremental approach to project management, emphasizing flexibility, collaboration, and rapid delivery of value. It embraces changing requirements and promotes adaptive planning. The Agile project life cycle typically includes phases like project initiation, iteration planning, iterative development, daily stand-ups, iteration review, and iteration retrospective.

3.2 Agile Manifesto: History of Agile and Agile Principles: The Agile Manifesto is a guiding document for Agile software development. It emphasizes individuals and interactions, working software, customer collaboration, and responding to change. The history of Agile can be traced back to the early 2000s when a group of software practitioners came together to define a more flexible and customer-centric approach to software development.

3.3 Key Agile Concepts: 3.3.1 User Stories, Story Points: User stories are concise descriptions of a software feature from an end-user's perspective. They capture the user's needs and help in defining the scope of the software project. Story points are a relative estimation technique used in Agile to estimate the effort required to implement a user story. They provide a measure of complexity and effort without specifying the exact time required.

3.3.2 Product Backlog: The product backlog is a prioritized list of user stories or features that need to be implemented in the software product. It serves as a dynamic document that evolves over time as new requirements emerge or existing ones change. The product backlog helps in managing scope and prioritizing work.

3.3.3 Sprint Backlog: The sprint backlog is a subset of user stories or tasks selected from the product backlog for a specific sprint. It represents the work that the Agile team commits to completing within the sprint. The sprint backlog is created during the sprint planning session and serves as a guide for the development team during the sprint.

3.3.4 Sprint Velocity: Sprint velocity is a measure of the amount of work a team can complete within a sprint. It is calculated by summing up the story points or effort of the completed user stories in a sprint. Sprint velocity helps in predicting how much work a team can accomplish in future sprints.

3.3.5 Swimlanes: Swimlanes are visual representations used in Agile project management to organize and track work. They divide the project board or Kanban board into horizontal lanes, each representing a specific category or team responsible for completing certain tasks. Swimlanes provide clarity and facilitate better workflow management.

3.3.6 Minimum Viable Product (MVP): The Minimum Viable Product (MVP) is the smallest version of a product that contains the core features and provides value to the end-users. It is developed and released early to gather feedback and validate assumptions. MVP helps in reducing development time, managing risks, and getting early customer input.

3.3.7 Version and Release: In Agile, software development is typically divided into iterations or sprints. At the end of each sprint, a version of the product is released, which includes the completed user stories or features. The release may be internal or external, depending on the project's requirements.

3.4 Agile Project Management vs. Traditional Project Management: Agile project management differs from traditional project management approaches, such as Waterfall, by promoting flexibility, adaptability, and iterative development. Agile focuses on delivering value incrementally, collaborating closely with customers, embracing change, and empowering self-organizing teams. Traditional project management, on the other hand, follows a linear and sequential approach, with a strong emphasis on upfront planning and documentation.

4. **Agile Teams, Size, and Schedule:**

4.1 Dynamic System Development Method: The Dynamic System Development Method (DSDM) is an Agile framework that provides principles, practices, and techniques for delivering software projects on time and within budget. DSDM emphasizes active user involvement, frequent delivery of products, and collaboration among stakeholders.

4.2 Value-Driven Development: Value-Driven Development focuses on delivering the maximum value to the customer by prioritizing features that provide the most business value. It involves continuous stakeholder engagement, early delivery of working software, and iterative development.

4.3 Team and Roles of an Agile Team: 4.3.1 Scrum Master: The Scrum Master is responsible for facilitating the Agile team's progress and ensuring adherence to Agile principles and practices. They act as a coach, removing obstacles, facilitating meetings, and promoting collaboration and self-organization within the team.

4.3.2 Product Owner: The Product Owner represents the stakeholders and is responsible for maximizing the value of the product. They define and prioritize the product backlog, collaborate with the team, make decisions on scope and requirements, and ensure alignment with customer needs.

4.3.3 Development Team: The Development Team consists of individuals with the required skills to deliver the product. They are self-organizing and cross-functional, responsible for implementing user stories, conducting tests, and delivering high-quality software increments.

4.4 Product Vision and Product Roadmap: The Product Vision is a high-level description of the desired future state of the product. It provides direction, purpose, and alignment for the Agile team. The Product Roadmap outlines the planned features and their prioritization over time. It helps in communicating the product's evolution and aligning stakeholders' expectations.

4.5 Project Objective and Key Metrics: Project Objectives define the specific goals and outcomes to be achieved through the project. They guide decision-making and provide a clear focus for the Agile team. Key Metrics are measurable indicators used to assess the project's progress and success. They can include metrics related to customer satisfaction, product quality, team performance, and business value delivered.

4.6 Introduction to User Stories: User Stories are concise, written descriptions of features from an end-user perspective. They capture the user's need, the feature's functionality, and its benefit. User Stories help in understanding requirements, prioritizing work, and facilitating collaboration between the Agile team and stakeholders.

4.7 Estimate the Product Backlog: Estimating the Product Backlog involves assigning effort or story points to each user story. The Agile team collectively estimates the relative complexity and effort required for implementing each story. These estimates help in prioritizing work, planning iterations, and forecasting project timelines.

4.8 Techniques for Estimating Story Points: Various techniques can be used to estimate story points, such as Planning Poker, T-Shirt Sizing, and Comparative Sizing. These techniques involve collaborative discussions and relative comparisons to assign story point values based on complexity, effort, and risk.

4.9 Plan Product Releases: Planning product releases involves deciding on the scope, content, and timing of the product increments to be delivered. It requires prioritizing user stories, considering dependencies, and aligning with stakeholder expectations. Release planning helps in setting expectations, managing customer feedback, and ensuring timely delivery of value.

4.10 Product Prioritization: Product Prioritization involves determining the order in which user stories or features should be implemented based on their business value, risk, and dependencies. Prioritization decisions are made collaboratively, considering stakeholder needs, market conditions, and project constraints.

5. **Tracking Agile Project and Reports:**

5.1 Introduction: Tracking Agile projects involves monitoring and controlling the progress of the project, identifying and addressing issues, and generating reports to communicate project status and metrics.

5.2 Plan and Execute Iteration: Planning and executing an iteration involve selecting user stories from the product backlog, estimating effort, creating a sprint backlog, and defining the tasks required to complete the selected user stories. The Agile team then works on implementing the user stories and tracking progress throughout the iteration.

5.3 Facilitate Retrospective, Making Team Decisions, and Closing out Retrospective: The retrospective is a meeting held at the end of each iteration to reflect on the team's performance, identify areas for improvement, and make decisions on how to enhance the Agile process. During the retrospective, the team discusses what went well, what could be improved, and action items for the next iteration. The retrospective is closed out by summarizing the key points and documenting the agreed-upon actions.

5.4 Agile Reports: 5.4.1 Daily Reports: Daily reports in Agile are short, focused status updates provided by each team member during the daily stand-up meeting. Each team member shares their progress, any challenges or blockers they are facing, and their plans for the day. Daily reports help the team stay aligned, identify and address issues quickly, and ensure transparency.

5.4.2 Sprint Burn-down Chart and Reports: The sprint burn-down chart is a visual representation of the work remaining versus time during the sprint. It shows the progress of the team in completing the committed user stories and helps track if the team is on track to deliver all the planned work. Sprint reports summarize the key metrics, progress, and any issues or risks encountered during the sprint. These reports provide insights into the team's performance and help in decision-making.

5.5 Benefits of Agile Project Management: Agile project management offers several benefits, including increased customer satisfaction through early and frequent delivery of value, adaptability to changing requirements, improved collaboration and communication among team members and stakeholders, faster time to market, and increased transparency and visibility into the project's progress.

6. **Implementation with Agile Tools:**

6.1 Introduction of Agile Tools: Agile tools are software applications designed to support Agile project management practices. These tools provide functionalities such as backlog management, task tracking, collaboration, reporting, and integration with version control systems. They help Agile teams streamline their processes, improve communication, and enhance productivity.

6.2 Hands-on GitHub: GitHub is a web-based platform widely used for version control and collaboration in software development projects. It offers features like project boards, repositories, issue tracking, and pull requests. GitHub can be leveraged as an Agile tool to manage tasks, track progress, and facilitate collaboration within an Agile team.

6.2.1 Create Project using Kanban: Kanban is a visual Agile framework that uses a Kanban board to visualize and manage work. In GitHub, you can create a project board with columns representing different stages of work, such as "To Do," "In Progress," and "Done." Tasks or user stories can be added as cards and moved across the board as they progress through the workflow.

6.2.2 Project Repositories: GitHub allows you to create repositories to store and manage your project's source code. It provides version control features, such as branching, merging, and code reviews. Repositories can be used to collaborate on code development, track changes, and ensure code quality.

6.2.3 Continuous Integration: GitHub supports continuous integration practices through integrations with various CI/CD (Continuous Integration/Continuous Deployment) tools. These integrations enable automated builds, testing, and deployment of the software, ensuring that changes are regularly integrated and tested.

6.2.4 Project Backlog: GitHub provides features to manage the project backlog. User stories or tasks can be created as issues and organized into milestones or epics. The backlog can be prioritized, assigned to team members, and tracked for progress.

6.2.5 Team Management: GitHub allows you to invite team members, assign roles and permissions, and manage collaboration within the team. Team members can be assigned tasks, communicate through comments, and receive notifications on updates.

6.2.6 Progress Tracking: GitHub provides visual indicators and progress tracking features. For example, you can track the progress of issues or tasks through labels, milestones, and assignees. The Kanban board provides a visual representation of the team's progress.

6.2.7 Releases: GitHub supports the management and tracking of software releases. You can create releases, tag specific versions of the code, and associate release notes or documentation. This helps in organizing and communicating the different iterations or versions of the software.

6.3 Implementation of Problem Statement with Agile Tools - GitHub: This case study involves using GitHub as an Agile tool to manage an Agile project. It includes designing the product vision, creating the product backlog with features and user stories, estimating story points, designing the iteration plan, tracking iteration progress, and closing the iteration. Screenshots can be added with captions to illustrate the process and usage of GitHub as an Agile tool.