Learning Journal

Name :Ameer Sohail Mohammed
ID : 40230877

SOFTWARE PROJECT MANAGEMENT

1. Conceptual Model and Software Engineering Management: The lecture mentions that a conceptual model can reveal more detailed aspects of software engineering management. It highlights the challenges of dealing with technology changes and the need for insulation against variability in technology.

2. Software Engineering Management: Model II: The lecture briefly mentions Model II, which is a conceptual framework for effective software engineering management. It emphasizes the importance of avoiding overpromising and discusses a proposed conceptual framework of manager attributes.

3. Agile Methodologies in Software Project Management: The lecture states that agile methodologies have brought notable changes to software engineering and have given rise to agile project management. It mentions various sources and authors related to agile methodologies.

4. Characteristics of Software Projects: The lecture describes the four classes of software projects: simple, complicated, complex, and chaotic. It explains the level of uncertainty and challenges associated with each class.

5. Principles of Software Project Management: The lecture mentions several principles, including the importance of clear goals, prioritizing quality, and the role of disciplined and motivated people in developing quality software. It also discusses the principle of fixed point of maximum compressibility in software project schedules.

6. Change Management and Requirements Management: The lecture highlights the need for change management and requirements management in software projects. It mentions the challenges of software requirements change and the importance of reactive, adaptive, iterative, and recursive software engineering processes.

7. Software Project Management Maturity Models: The lecture mentions the existence of various proposals for software project management maturity models. It

refers to different authors and organizations that have developed these models over the years.

8. Software Engineering Management Tools: The lecture briefly mentions the selection and use of tools to manage software engineering projects as a sub-area of software engineering management.

These topics provide an overview of software project management, including conceptual models, agile methodologies, characteristics of software projects, principles of project management, change management, requirements management, maturity models, and the use of management tools.

SOFTWARE PROJECT ASSESSMENT

Topic: Classification of Critical Success Factors for Software Projects

- There are different classification schemes for critical success factors in software projects.
- Classification Scheme 1: In a survey of literature, 26 critical success factors are identified and classified into People Factors, Process Factors, and Technical Factors.
- Classification Scheme 2: A chronological summary of initiatives that outline critical success factors for software projects is provided.
- Classification Scheme 3: A conceptual model of critical success factors is given, classified into Communication Factors, Technical Factors, Organizational Factors, Environmental Factors, Product Factors, Team Factors, and Project Management Factors.
- Other possible classification schemes include Primary and Secondary.
- The majority of critical success factors are non-technical factors.

Topic: Concerns and Issues in Software Project Reports

- Specific concerns in software project reports include implicit definition of success, questionable criteria of success, unavailability of source data, lack of explicit methodology, overly alarmist tone, and biased or misleading results.
- Some reports' conclusions have been confirmed by other surveys.

Topic: Assessments and Studies of Software Project Outcomes

- Assessments of software project outcomes, specifically failures, have been studied analytically and empirically.
- Empirical studies often use surveys as instruments.
- Various organizations conduct surveys to assess software project outcomes.

Topic: Success Criteria/Success Factors for Software Projects

- Success criteria or success factors for software projects are outlined in different sets.
- The criteria/factors are independent of application domain, development methodology, and implementation language.
- The criteria/factors are derived from surveys, but the underlying rationale and pre-defined acceptable values are not always provided.

Topic: Estimation Challenges in Software Projects

- Estimation techniques/tools for cost and duration in software projects are often lacking or considered poor.

Topic: Challenges in Software Project Collaboration

- Challenges in software project collaboration include convergence of people from different disciplines, lack of mutual understanding of technology, conflicting personalities, hidden political agenda, ineffective meetings, lack of proximity, poor infrastructure and support, self-proclaimed experts unwilling to listen, overwhelmed by size or complexity, and lack of good communication structure.

Stakeholder Management

1. Importance of studying people: The lecture emphasizes the significance of studying people, particularly in the context of software engineering projects. It references a book titled "Peopleware" by DeMarco and Lister to underscore this importance.

2. Stakeholders in project management: The Project Management Body of Knowledge (PMBOK) and the Software Engineering Body of Knowledge (SWEBOK) discuss the concept of stakeholders. The PMBOK provides a definition

of stakeholders, while SWEBOK identifies and defines stakeholders in various sub-areas of the Software Requirements Knowledge Area.

3. Limitations of stakeholder coverage: Both PMBOK and SWEBOK are criticized for their inadequate coverage of stakeholders. They lack comprehensive treatment of stakeholder involvement and fail to address relationships between stakeholders.

4. Organizational charts and stakeholders: Organizational charts depict the hierarchy within an organization but may not include external stakeholders or illustrate their relationships. This can create a perception that stakeholders operate in isolation.

5. Understanding the value of software projects: Stakeholders play a crucial role in determining the value of a software project. They contribute to the project's value by considering aspects such as people, process, and product.

6. Planning fallacy and overscoping: Not knowing important features of a project can lead to overscoping. The lecture suggests using the web to understand the concept of "Planning Fallacy," which relates to the tendency to underestimate the time and resources required for a project.

7. Contextual knowledge and stakeholders: Acquiring contextual knowledge about software systems can be challenging. The lecture mentions that subscribers of gaming magazines, gaming event attendees, and active participants in gaming communities can be potential stakeholders for a computer game project.

8. Stakeholder identification and requirements elicitation: Understanding stakeholders is crucial for effective requirements elicitation. Incomplete knowledge of stakeholders can lead to incomplete requirements. The identification of stakeholders is considered a good practice in requirements engineering.

9. Model of software engineering: The IEEE Software and Systems Engineering Standards Committee provides a model of software engineering that includes various interrelated elements, with people being one of them.

SOFTWARE PROJECT COST ESTIMATION

Topic: Introduction to Software Project Cost Estimation
- Accurate estimation is important for software project success.
- Consideration of a large number of irrelevant parameters is often infeasible due to high costs.

- Example: It may not be economically viable to consider all cost drivers if a few account for a significant portion of observed productivity variability.

Topic: Risk Analysis
- Risk analysis involves assessing future events with known possible outcomes, some of which are negative.

Topic: Estimation Models
- There are various cost estimation models available for software development.
- Cost estimation models can be classified in different ways.

Topic: Limitations and Considerations
- The definition of terms like "month" can vary across regions and organizations and should be defined at the outset.
- There are limitations to current approaches for cost estimation.
- Cost estimation is both a science and an art, and objective estimation is usually not possible.

Topic: Planning and Trade-offs
- Estimation of cost helps in making trade-offs in software projects.
- Reusable entities of knowledge, such as APIs and class libraries, may not have a price tag, leading to trade-offs in cost estimation.

Topic: Relationship between Size and Effort
- The relationship between size and effort in software projects is not always clear.
- Linear relationship between size and effort should not be expected, and there can be different kinds of nonlinear relationships.

Topic: Planning Onion
- Planning can be organized in interrelated layers, such as Release, Iteration, and Daily planning in agile software projects.

Topic: Probability and Interval
- Probability is meaningful only for intervals of non-zero length, not for specific points.

Topic: Historical Context
- The lecture references historical quotes and observations related to estimation and exponential growth.

Topic: Regression-Based Cost Estimation

- Regression-based cost estimation equations can be used to estimate software project costs.

SOFTWARE PROJECT TEAMS

Topic: Bloom's Taxonomy

- Bloom's Taxonomy is a hierarchical model used to classify skills and behavior important to learning.
- It consists of three domains: cognitive (knowledge-based), affective (emotive-based), and psychomotor (action-based).
- Each level of the taxonomy involves different cognitive processes, such as remembering, understanding, applying, analyzing, and evaluating.

Topic: Leadership Principles

- General principles of leadership are important for effective team management.
- Figure 9 lists certain general principles of leadership, along with those who are affected by those principles.
- The team leader plays a vital role in managing the team, facilitating communication, and resolving conflicts.

Topic: Software Engineering Body of Knowledge (SWEBOK)

- The SWEBOK has a number of Knowledge Areas (KAs), one of which is Software Engineering Professional Practice (SEPP).
- Figure 15 shows the Software Engineering Professional Practice Knowledge Area of SWEBOK.

Topic: Team Configuration in DevOps

- In DevOps, certain team topologies are encouraged while others are discouraged.
- Figure 4 illustrates the recommended team topologies in DevOps.

Topic: Team Composition and Skills

- It is important for the composition of a team to reflect an adequate diversity of skills beyond programming.

- Skills such as critical thinking, lectureing, modeling, negotiating, and testing are necessary for a successful software project.

Topic: Cognitive Biases in Software Engineering Decision Making

- Various cognitive biases can affect decision making in software engineering.
- Examples of cognitive biases include anchoring bias, availability bias, confirmation bias, and overconfidence effect.

Topic: Team Size

- The optimal team size for software projects is typically 5 members.
- A difference in team size can impact communication complexity and should be minimized.

Topic: Responsibilities of a Team Leader

- The team leader in software engineering course projects has responsibilities such as managing the team, facilitating communication, and ensuring project success.

Topic: Classification of Teams

- Teams can be classified based on their intent and the things they support.
- Table 1 shows different types of teams based on their intent.

Topic: Communication Complexity in Teams

- The rate of increase in communication complexity in a team is a function of the team size.

Topic: Equivalence Class, Permutation, and Combination

- Equivalence class refers to a set of inputs that produce the same output in software testing.
- Permutation and combination are different mathematical approaches used to represent team configurations.

Topic: Unanticipated Changes in Software Projects

- Unanticipated changes, such as changes in technology or tools, can be challenging in software projects.
- Stability of a team over the term is important, but not guaranteed, and changes in team membership should be communicated to the team leader and teacher/professor.

Topic: Individual and Social Skills in Teams

- Certain individual and social skills are desirable in team members, including the team leader.
- These skills go beyond the realms of any project, course, or program.

Topic: Optimal Team Size

- After supervising many software project teams, it has been found that a team of size 5 is optimal.

Topic: Omission of Women in Computer Science

- The omission of women from the history of computer science perpetuates misconceptions and stereotypes.
- It is important to include and recognize the contributions of women in the field.

Topic: Cognitive Biases in Decision Making

- Various cognitive biases, such as anchoring bias and confirmation bias, can impact decision making in software engineering.

Topic: Four Stages of Learning Model

- The Four Stages of Learning provides a model for learning, with hierarchically organized stages.
- The stages include unconscious incompetence, conscious incompetence, conscious competence, and unconscious competence.

SOFTWARE RISK MANAGEMENT

Topic: SWEBOK and Risk Management
- SWEBOK places special emphasis on risk in software engineering economics.
- Risk and Uncertainty is a sub-area of the Software Engineering Economics Knowledge Area in SWEBOK.

- Risk management is covered differently in different disciplines, both inside and outside project management.

Topic: Classification of Risk
- Different types of risk classes have been identified, such as Software Requirements Risks, Software Technical Risks, Software Schedule Risks, Software Project Risks, Software Supplier Risks, and Software People Risks.
- A taxonomy of software risks provides a low-level view of different types of risks in software projects.

Topic: Identifying Risks
- Risk identification is an important step in risk management.
- The details of risk identification processes are given in Figure 10.

Topic: Limitations, Weaknesses, Vulnerabilities, Threats, and Risks
- Imperfections or limitations in a software project can become active weaknesses depending on their role and context.
- Vulnerabilities, such as organizational culture or inadequate support for user interface design, can be exploited by threats.

Topic: Project Risk Management
- Project Risk Management is one of the Knowledge Areas (KAs) covered in the IEEE Standard 1490.
- It consists of processes like Plan Risk Management, Perform Qualitative Risk Analysis, Perform Quantitative Risk Analysis, Plan Risk Responses, and Monitor and Control Risks.

Topic: Risk Modeling
- Risk modeling is a method used to bring order to chaos in complex systems.
- It can be used in software projects to manage risks.

Topic: Clear Understanding of Risk Management Concepts
- It is important to have a clear and consistent understanding of risk management concepts across different disciplines.
- Risk management is covered differently in different disciplines, both inside and outside project management.

Topic: Assumptions in Project Plans
- Project plans often contain assumptions about commitments.

- It is crucial to have means to prepare for and control situations where these assumptions turn out to be wrong.

Topic: Global Software Development Risks
- Geographical, socio-cultural, and temporal distances in global software development can pose various risks.
- Different interpretations of concepts and limited face-to-face meetings can be challenges in global software development.

In lecture, cover the importance of risk management in software development projects, the classification of risks, risk identification processes, vulnerabilities and threats, project risk management processes, risk modeling, understanding risk management concepts, assumptions in project plans, and risks in global software development.

EARNED VALUE MANAGEMENT

1. Introduction to EVM:
   - EVM determines the progress of a project by measuring the performance of individual tasks and overall project performance.
   - It is important for project management as it allows project managers to monitor progress, control budget and schedule, and make informed decisions.
   - EVM is supported by organizations such as NASA and the Project Management Institute.

2. EVM Metrics and Reporting:
   - EVM metrics are interdependent and used for status reports, progress reports, forecasts, and variances.
   - The metrics provide insights into the project's current status, predicted outcomes, and deviations from the planned budget and schedule.
   - Different stakeholders may be interested in different metrics.

3. EVM in Agile Projects:
   - EVM can be applied to agile projects, with metrics calculated for each iteration.
   - Agile-specific notions such as story points and user stories replace conventional notions like effort and task.
   - Initiatives like AgileEVM and AEVMS have extended EVM for agile projects.

4. EVM and Project Recovery:

- EVM metrics can be used to facilitate project recovery by identifying areas for improvement.
- If a project is under budget and behind schedule, overtime work can be considered to complete more tasks.

5. Limitations of EVM:
- EVM is not applicable to projects without detailed planning or those based on extreme agile methodologies.
- It does not measure project scope or product quality, focusing primarily on budget and schedule.

6. Relevance of EVM for Project Managers:
- Project managers need to monitor and control project characteristics, including time and money.
- EVM allows proactive monitoring and control of budget and schedule, contributing to project success.

7. EVM in Software Engineering:
- EVM is relevant to software engineering economics and is covered in the SWEBOK.
- It is part of the Project Cost Management Knowledge Area in the IEEE Standard 1490.

8. Support for EVM:
- EVM is supported by organizations such as NASA and the Project Management Institute.

This lecture provides an overview of the key aspects of EVM, including its importance, metrics, application in agile projects, project recovery, limitations, relevance for project managers, and its relationship with software engineering.

RETURN ON INVESTMENT

1. Introduction to ROI:
- ROI (Return on Investment) is a measure of the profitability of an organization.
- It is a central topic in software engineering economics.

2. Classification of ROI:
- There can be different kinds of ROI, such as financial ROI and schedule ROI.

- Financial ROI focuses on cost savings, while schedule ROI focuses on schedule savings.

3. ROI Models:
   - ROI Model I: ROI1 = (Cost Saved – Cost Consumed) / Cost Consumed.
   - ROI Model II: ROI2 = (Cost Saved – Cost Consumed) / Initial Cost.

4. ROI Calculation Examples:
   - Example 1: ROI1 = (100 – 10) / 10 = 9.
   - Example 2: ROI2 = (1000 – 910) / 1000 = 0.09 (Project A), ROI2 = (1000 – 460) / 1000 = 0.54 (Project B).

5. ROI at Project Level:
   - The lecture focuses on ROI at the software project level, rather than at the organizational level.
   - Organizational level ROI analysis aggregates the results of ROI analysis for all projects within the organization.

6. Relationship between ROI and Software Development Methodologies:
   - The selection of software development methodology can affect ROI.
   - Rigid methodologies like the Waterfall Model and agile methodologies like Scrum have different timeframes for ROI.

7. Limitations of ROI Model I:
   - ROI1 does not accurately account for the benefits of investments in software projects.

8. Tools and Metrics for ROI:
   - Various models and tools can be used to evaluate financial ROI.
   - EVM (Earned Value Management) metrics can be used for project recovery.

This lecture provides an overview of the key aspects of ROI, including its definition, classification, models, calculation examples, project-level analysis, relationship with software development methodologies, limitations, and tools/metrics for ROI evaluation.